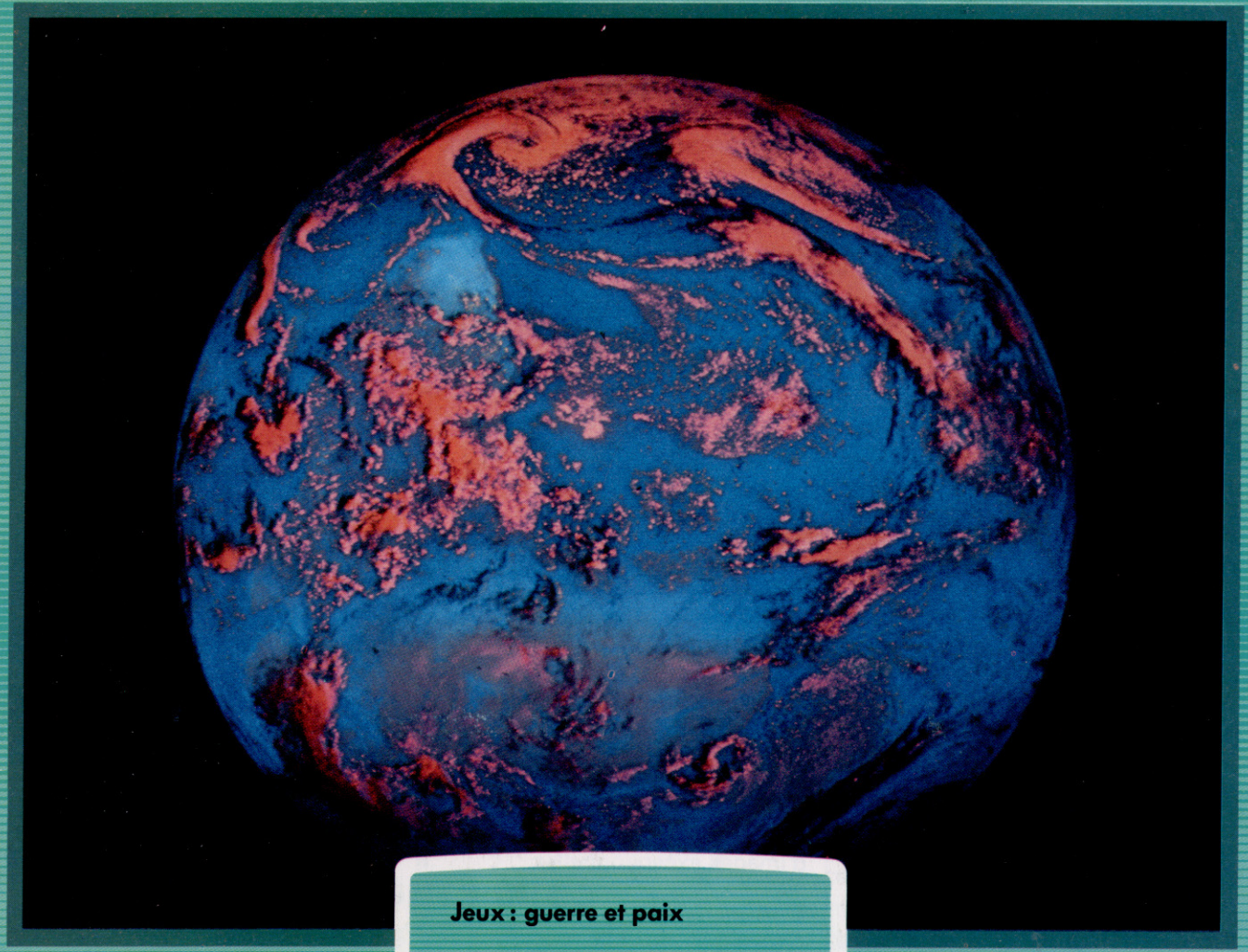
ABC INFORMATIQUE

N° 23

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Jeux : guerre et paix

Reconnaissance de la voix

Code machine

Le QL Sinclair

EDITIONS ATLAS

M6062-23-12F 85FB-3,80FS-$1.95

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15.* Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15.* Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume II, n° 23.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Matra.

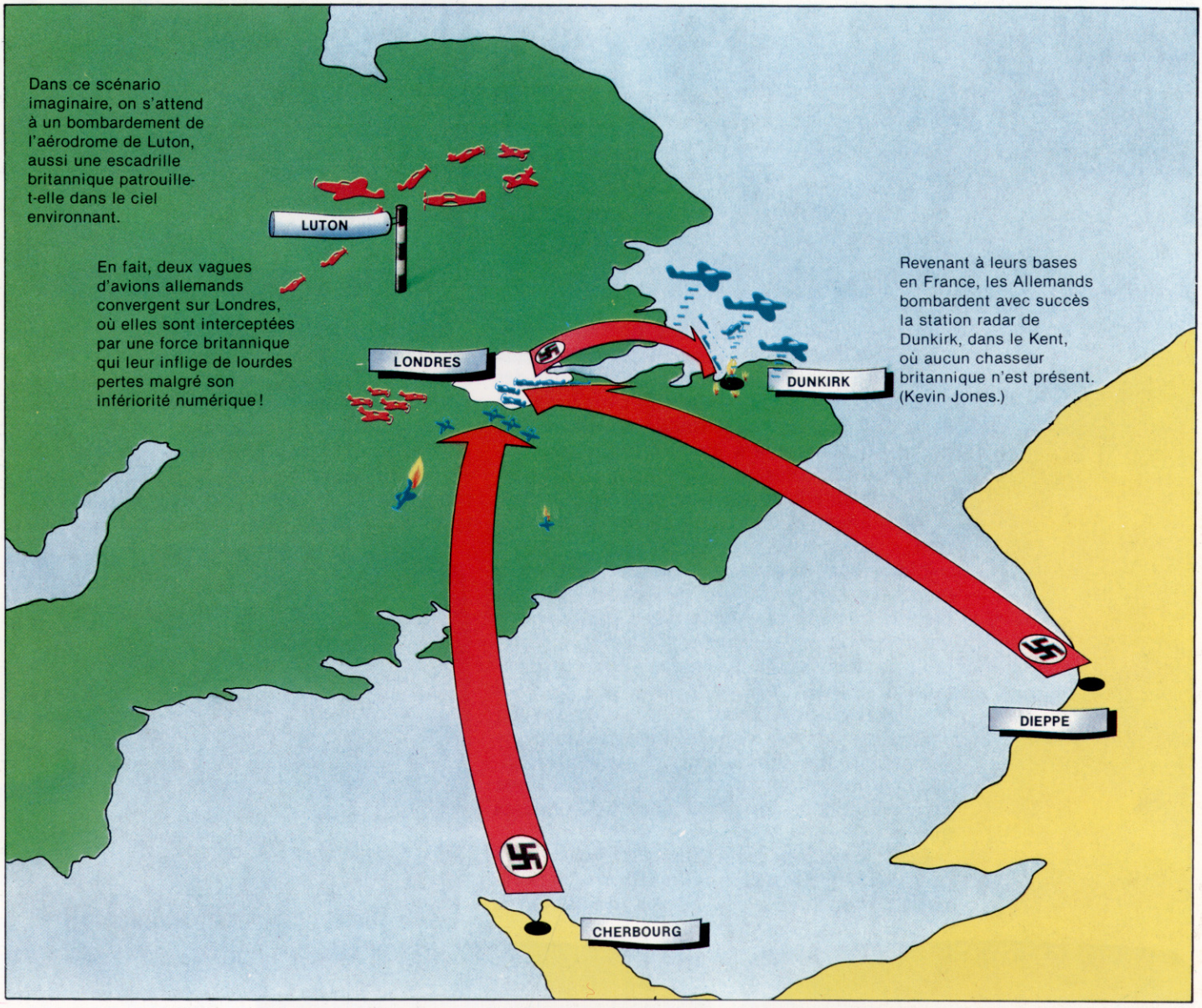
Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juin 1984. 8846. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

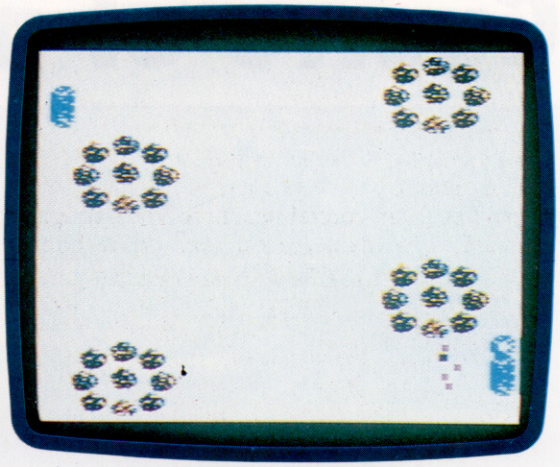
Guerre et paix



Vous pouvez vous procurer des jeux qui vous permettent de tester vos talents de stratège et de tacticien militaire dans des reconstitutions de batailles.

Les responsables militaires d'aujourd'hui accordent une grande importance aux jeux de stratégie, les *war games*. Ils n'hésitent pas à y jouer afin d'éprouver certaines ripostes en cas de menace ou d'attaque surprise. Ces jeux sont très élaborés. Il a fallu mettre au point des matériels et des logiciels capables de reproduire tous les aspects possibles d'un conflit : déploiement initial des forces alliées et ennemies, ravitaillement, réserves, et ainsi de suite. On peut aussi prévoir des conditions atmosphériques défavorables, des modifications de la tactique de l'adversaire, la présence d'une « cinquième colonne », bref toutes les variables qui peuvent

La Bataille d'Angleterre
Fighter Command (prévu sur Apple) est un parfait exemple de ces jeux stratégiques sophistiqués disponibles sur micro-ordinateurs. Avant de commencer, le joueur doit sélectionner un certain nombre de variables, qui vont du type d'avion utilisé aux conditions météorologiques. L'affichage se fait par déplacement de symboles sur la carte, où vient aussi s'imprimer un texte qui comporte des renseignements supplémentaires. (Cl. Ian McKinnell.)



Tactical Armour Command
Ce jeu est disponible sur divers micros comme Apple, Atari ou Commodore 64. Il reproduit des combats de blindés durant la Seconde Guerre mondiale. On peut y jouer seul ou à deux; il y a cinq scénarios différents, qui font s'affronter les forces des armées alliées et celles des armées allemandes. (Cl. Ian McKinnell.)

affecter le déroulement d'une opération militaire. Une des fonctions essentielles du système de défense radar NORAD, installé au milieu des Cheyenne Mountains, aux États-Unis (vu dans le film *War Games*) est d'établir, d'évaluer et de mettre constamment à jour les forces respectives des États-Unis et de l'Union soviétique, ainsi que de contribuer à la mise au point de ripostes adaptées à toute évolution de la situation.

L'amateur dispose de moyens beaucoup moins sophistiqués. Pour recréer un affrontement dans toute sa complexité, il a dû recourir longtemps à de volumineux règlements, à des tables, à des dés mille fois lancés. Cela représentait un travail énorme. Seule une poignée d'enthousiastes acceptait d'en passer par là. Toutefois, depuis l'avènement du micro-ordinateur, des programmes de jeux stratégiques permettent de s'épargner des tâches aussi fastidieuses et ils deviennent aussi passionnants que difficiles.

Leur variété est par ailleurs immense : on peut recréer ou simuler pratiquement n'importe quel affrontement, des guerres de la Grèce antique au possible conflit entre les pays de l'O.T.A.N. et ceux du pacte de Varsovie. On peut se battre dans les airs, sur les mers, dans l'espace, ou même opposer des empires mythologiques. Le choix proposé aux joueurs est à peu près sans limite.

Les jeux historiques vous permettent de déterminer quand Napoléon a commis des erreurs à Waterloo, ou de barrer le chemin à Hitler lors de son invasion de l'Union soviétique en 1941. Les jeux galactiques offrent encore plus de possibilités d'invention. Vous devez non seulement manœuvrer vos flottes de combat à travers la galaxie, mais encore préciser quels types de vaisseaux vous désirez. Il vous faudra faire des compromis : un gain de vitesse n'est possible qu'aux dépens de la capacité d'armement, et des champs de forces protecteurs réduisent la part allouée au carburant. Les choix que vous serez amené à faire dépendront de votre style de combat.

Légionnaire
Cette recréation des guerres entre les forces de César (vous) et celles des Barbares (l'ordinateur, Apple ou Atari) est jouée en temps réel. L'infanterie, la cavalerie et les forces annexes sont représentées par des signes conventionnels, et peuvent être déplacées grâce à un curseur (c'est le carré blanc) contrôlé par un manche à balai. (Cl. Ian McKinnell/Soft.)

Les jeux stratégiques n'exigent que peu d'habileté et de connaissances particulières, contrairement à ce qui se passe sur le terrain; la plupart sont accompagnés de notes et de conseils destinés aux débutants. Toutefois, certains sont plus difficiles que d'autres. On distingue les jeux pour débutants, pour joueurs déjà avertis et pour connaisseurs. Si vous songez à aborder ce domaine, il est sans doute bon de commencer par le début et de vous familiariser avec les règles de base grâce à des jeux simples, avant de passer au niveau supérieur.

Les combats mis en scène sont de types très divers, et le format même du jeu varie à chaque fois. De façon générale, la partie se déroule sur une grande carte; lorsqu'elle est trop vaste pour tenir tout entière sur l'écran, celui-ci joue le rôle d'une « fenêtre » qui se déplace à volonté au moyen d'un manche à balai ou d'une souris. Dans le cas d'une bataille historique, il faut bien entendu que le programmeur reconstitue, aussi fidèlement que possible, le cadre réel où s'est déroulée l'action. Quand l'affrontement prend place essentiellement dans l'Atlantique Nord, cela ne pose guère de problèmes de graphisme; mais, dans un jeu comme « la Bataille de Normandie », on cherche à recréer le débarquement du 6 juin 1944, ce qui est infiniment plus difficile : les plages, les côtes, les villages, les rivières, la topographie générale, tout cela doit être rigoureusement authentique. Même dans la description d'affrontements imaginaires, il faut apporter au jeu suffisamment de variété, d'épreuves et surtout d'équilibre, afin qu'aucun des deux camps n'ait la tâche plus facile que l'autre.

Une grille vient généralement se superposer à la carte, qu'elle divise un peu comme un échiquier, à ceci près que les subdivisions sont très souvent hexagonales et non carrées. Chacune d'elles reçoit une certaine valeur, qui est fonction du terrain qu'elle représente : elle est d'autant plus grande que ce terrain est difficile à traverser. Lorsqu'une unité particulière entreprend de s'y mouvoir, sa liberté de mouvement est amputée de cette valeur. Elle ne peut plus se déplacer si cette liberté est égale à zéro, ou si elle est déjà inférieure à la valeur de la zone où elle veut entrer.

Ordinairement chaque joueur joue à son tour, en un temps limité, au cours duquel il doit réussir à atteindre un certain nombre d'objectifs s'il veut remporter la partie. La plupart du temps, il n'est pas nécessaire (ou pas possible) de tout faire. La première décision consiste donc, après évaluation des chances, à définir un ordre de priorités stratégiques. Le rôle de l'adversaire consiste souvent à empêcher l'attaquant d'atteindre son but. Mais lui non plus ne peut défendre toutes ses positions. Certaines sont condamnées et il doit les abandonner; d'autres sont plus solides et il faut les tenir. Il doit aussi chercher à voir s'il peut prendre le risque de lancer des contre-attaques pour regagner le terrain perdu, ou pour gêner les préparatifs d'une future offensive.

Le programme permet un affichage graphique des combats (représentation du terrain, forces en présence), à quoi viennent s'ajouter divers renseignements textuels : efficacité des unités de combat et liberté de mouvement de chaque unité. Le joueur choisit l'une d'elles en déplaçant un curseur sur une liste, ou c'est l'ordinateur qui les lui présente l'une après l'autre; il prend alors une décision. Cela exécuté, l'ordre de déplacement est donné. Si l'on prend le cas d'une grille à cases hexagonales, 1 permet d'envoyer l'unité au nord, 2 au nord-est, et ainsi de suite en respectant l'ordre de succession des directions.

Mais de plus en plus de jeux fonctionnent à l'aide d'un manche à balai ou de poignées de jeux; le déplacement est alors beaucoup plus facile et rapide, car la manipulation se fait directement. La commande FINISH (en abrégé F) signale que le déplacement est achevé. Mais souvent, le joueur peut mettre par la suite la même unité en mouvement, s'il le désire et si elle dispose encore d'une certaine liberté de mouvement. Puis on avertit l'ordinateur en tapant la commande EXECUTE (ou E). Il lancera alors la phase de combat.

Le front de l'Est
Dans ce jeu, le joueur contrôle l'armée allemande qui s'efforce d'atteindre Moscou en 1941; l'ordinateur dirige les forces soviétiques. Rédigé par Chris Crawford et distribué par Atari, le programme comporte certaines nouveautés comme la possibilité d'un défilement très fin de l'écran. Un des grands attraits visuels du jeu réside dans le changement de couleur de la carte en fonction des saisons. (Ian McKinnell/Soft.)

Durant toute cette phase, il signalera les unités amies en position d'engager la bataille, et donnera des informations sur les forces respectives. A partir de ces renseignements, le joueur peut ou non choisir l'affrontement, suivant les possibilités offertes. Les résultats de toute opération sont calculés par la machine, puis affichés; c'est alors au second joueur de s'asseoir devant l'ordinateur.

Pour beaucoup de gens, la fascination des jeux stratégiques tient à ce qu'il n'y a pas de solution « correcte » aux problèmes posés. Il faut à la fois surmonter les difficultés du terrain, les problèmes logistiques, et employer au mieux les ressources dont on dispose pour vaincre l'adversaire : c'est une sorte de défi intellectuel. Bien entendu, chacun aimerait l'emporter grâce à des plans audacieux, à des pièges soigneusement tendus, mais ce qui compte avant tout, c'est la victoire!

Les utilitaires

Les « ensembles utilitaires » sont destinés à renforcer le basic, et à aider le programmeur à dépister ses erreurs.

Les premiers micro-ordinateurs (Apple II, Pet, Commodore, etc.) avaient des performances limitées. Ils manipulaient des nombres et du texte. Le BASIC, destiné à ces machines, fournissait des routines et des commandes réduites. De nombreux ensembles utilitaires furent donc écrits, habituellement en code machine. Ils intervenaient en dehors du BASIC en fournissant une aide à la programmation sous la forme de commandes pour la construction des programmes et la recherche des erreurs.

Les ingénieurs ont développé, depuis, de nombreuses possibilités graphiques et sonores, suivant en cela le succès croissant des jeux sur micros. Chaque nouveau modèle apporte de nouvelles caractéristiques qui sont aussitôt incorporées aux logiciels professionnels.

Utilitaires de travail
Voici quelques utilitaires et extensions BASIC parmi les plus répandus. Les ensembles utilitaires qui permettent la création de figures graphiques sur des micros qui en sont normalement dépourvus deviennent très recherchés.

Ensembles utilitaires	
SUPER TOOL KIT	Pour les Spectrum 16 et 48 K
SPECTRUM EXTENDED BASIC	Pour le Spectrum 48 K
SPECTRUM KEY DEFINE	Pour le Spectrum 48 K
PROGRAMMER'S AID	Pour le Vic-20, produit par Commodore
BUTI	Pour le Vic-20
TOOL BOX	Pour les modèles A et B du BBC, par BBC Software
SPRITE MAGIC	Pour le Dragon 32
SPRITE GRAPHICS	Pour le Spectrum 48 K
SPRITE MASTER	Pour le BBC Model B

Cependant, le BASIC résident (à quelques exceptions près) n'a pas beaucoup progressé. Cela est dû précisément à ces « utilitaires », incorporés en tant que nouvelles commandes à l'aide de routines utilisant largement PEEK et POKE.

Il existe donc de nombreux ensembles utilitaires et de nombreuses extensions du BASIC, disponibles pour la plupart des machines. De manière générale, ils permettent un accès plus aisé à des caractéristiques déjà existantes (figures graphiques ou éditeurs de sons), étendent certaines commandes (création de figures graphiques) ou enfin, ils aident à la programmation en BASIC.

Ces extensions peuvent résider en RAM, en ROM interne, ou sur une cartouche ROM. Une extension ROM est préférable à une extension RAM dans la mesure où elle ne prend pas de mémoire utilisateur et où elle ne peut être effacée par erreur. D'une manière générale, un programme écrit avec l'aide d'un ensemble utilitaire ne pourra s'exécuter que sur ordinateur « configuré » comme l'ordinateur d'origine. Cependant, certains utilitaires créent des programmes indépendants de la configuration, comme les éditeurs graphiques et de figures, ainsi que les éditeurs de sons.

Les caractéristiques les plus intéressantes parmi les extensions BASIC sont les commandes graphiques spéciales (PAINT, DRAW, PLOT, CIRCLE...), les commandes son (SOUND, PLAY, MUSIC, ENVELOPE...), ou celles qui décrivent par leur nom un effet sonore (BANG, ZAP). Les commandes d'aide à la programmation structurée sont également très précieuses. Parmi celles-ci, REPEAT...UNTIL, et IF...THEN...ELSE. Ces instructions permettent à l'utilisateur d'écrire des programmes au déroulement logique, et lui évitent les maladresses d'un code saturé d'instructions GOTO.

Le basic Simon

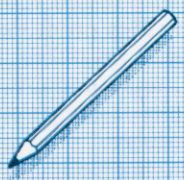
L'extension BASIC la plus complète est le BASIC Simon, destiné au Commodore 64 sous la forme d'une cartouche ROM. Le BASIC standard du Commodore 64 est resté très fruste, et dépourvu de commandes de programmation structurée. Bien que sa partie matériel soit très avancée (avec synthétiseur complet de sons, graphique haute résolution et figures graphiques), le contrôle BASIC de ces fonctions doit toujours se

Utilitaires machine

Commandes représentatives des meilleures caractéristiques d'une extension BASIC.

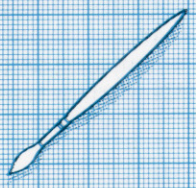
Des commandes utilitaires « disque » sont destinées au formatage de disque, à la copie et à la destruction de fichiers, à la copie de sauvegarde d'un disque.

DISK



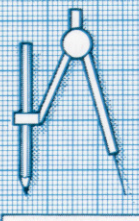
Outre les commandes destinées à tracer des lignes droites, en mode graphique haute résolution, DRAW permet le dessin point par point, ce qui est plus efficace pour établir des représentations graphiques.

DRAW



PAINT permet de colorier une zone. Sur certaines machines le curseur est au centre de la forme graphique et l'ordinateur la colorie en partant du centre, jusqu'à rencontrer une ligne.

PAINT

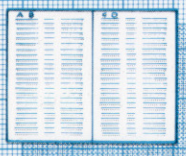


Les commandes graphiques telles que CIRCLE ont une fonction toute définie par leur nom, mais certaines permettent en outre le tracé d'arcs et d'ellipses.

CIRCLE

Les fonctions sonores varient d'une machine à l'autre, mais se rapportent toujours au jeu d'une séquence de notes définie comme une chaîne.

MUSIC



Le *DIRectory* (répertoire) tient un état des noms, types et tailles des fichiers présents sur un disque. La commande DIR vous permet de visionner le répertoire des fichiers, sans perdre le programme en cours en RAM.

DIR

La commande JOYstick (manette de jeu) donne la position de la manette aux variables BASIC, sans passer par PEEK et POKE.
(Kevin Jones.)

JOY

faire par PEEK et POKE. L'extension BASIC de Simon élargit considérablement le champ du BASIC Commodore, en lui ajoutant les caractéristiques suivantes :

1° Un ensemble complet d'aides à la programmation, comprenant de nouvelles fonctions de contrôle du listage, des commandes pour le dépistage d'erreurs et pour la protection de vos programmes (contre toute reproduction).

2° De nouvelles commandes de manipulation de chaînes et de texte.

3° D'autres opérateurs arithmétiques et des commandes de conversion numérique.

4° Des commandes simplifiées de manipulation de disques.

5° Des commandes graphiques haute résolution qui permettent de mélanger texte, tracé de points et formes géométriques. Les contours peuvent aussi être coloriés.

6° Des commandes graphiques basse résolution et des commandes sur l'écran qui peuvent copier et manipuler facilement chaque zone des images. On peut également recopier l'écran sur disque, bande ou imprimante.

7° Créateur et éditeur de figures graphiques facilement utilisable.

8° Commandes de procédures destinées à la programmation structurée, telles que PROC, CALL et EXEC. Routines également de contrôle de boucles et de conditions logiques, comme REPEAT...UNTIL, LOOP...EXIT, IF...END LOOP; enfin IF...THEN...ELSE, les routines qui éliminent les instructions GOTO et GOSUB.

9° Routines de création de sons permettant l'accès aux 64 possibilités sonores, par l'intermédiaire de simples commandes modulant le son et le jeu.

10° Commandes simples d'utilisation du crayon optique, de la manette de jeu et de la manette de commande.

Il est rare qu'une extension BASIC comporte autant de routines supplémentaires. La plupart des utilitaires ne s'intéressent qu'à un seul aspect de la programmation. Ainsi, la cartouche Super Expander qui permet de passer du Commodore au Vic-20 ne fournit que des commandes graphiques haute résolution et des commandes sonores. Les utilitaires les plus recherchés sont ceux qui concernent l'aide à la programmation structurée. Leurs commandes sont sur une seule clef, et ils fournissent des routines automatiques qui simplifient la numérotation des lignes, l'édition, la recherche des erreurs, le tout en mode direct.

Les utilitaires et les extensions BASIC permettent une meilleure exploitation du BASIC résident. Les routines qui améliorent les performances des micros sont plus rares. Certains utilitaires, néanmoins, apportent réellement de nouvelles possibilités, dans la création de figures graphiques pour l'écriture de jeux d'exécution rapide par exemple.

Les utilitaires BASIC, ensembles utilitaires et les extensions BASIC que nous avons décrits ne sont qu'une petite fraction des utilitaires et des programmes d'aide existants.

Les voix du Seigneur

Les dispositifs de reconnaissance de la parole sont déjà employés dans certains domaines comme le commerce ou la sécurité. Mais leur puissance reste limitée par la capacité mémoire de l'ordinateur.

Pour qu'un ordinateur puisse servir à quelque chose, il lui faut un moyen de recevoir et de transmettre des informations. Le procédé le plus employé demeure le clavier (bien qu'on puisse aussi recourir à une souris ou à un manche à balai). Mais l'usage du clavier oblige à communiquer avec l'appareil par le biais d'un langage artificiel. Des commandes comme CLS, DIRECTORY, RUN, LOAD ou SAVE ont bien entendu un sens pour le système d'exploitation; elles n'en sont pas pour autant « naturelles ».

Si l'ordinateur pouvait comprendre des commandes prononcées à voix haute — même si elles étaient construites comme celles qui sont tapées sur le clavier —, il serait d'usage bien plus facile, surtout pour les handicapés. Toutefois, avant de reconnaître des mots, il doit d'abord traiter les sons. Ces sons, qui sont des signaux analogiques, doivent en premier lieu être analysés et traduits sous forme numérique avant que l'ordinateur puisse les étudier. Cela paraît à première vue facile; mais la parole est une combinaison de sons extrêmement complexe.

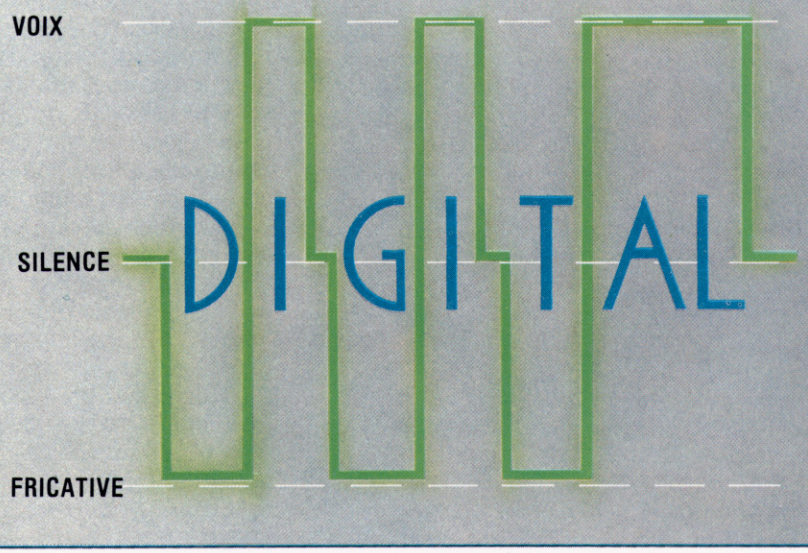
Un système de reconnaissance immédiate et exhaustive de la parole (comme dans *2001, l'Odyssée de l'espace,* avec l'ordinateur Hal) appartient encore, et pour longtemps, au domaine de la science-fiction; de même que la machine à écrire commandée à la voix. A ce stade, la technologie existe déjà, mais intervient une question de prix : tout cela coûte beaucoup trop cher. Les systèmes de reconnaissance de la parole doivent surmonter une difficulté fondamentale : les mots peuvent se prononcer de la même façon tout en ayant des sens différents. La puissance de traitement, seule capable de résoudre ce problème, n'est tout simplement pas disponible à un prix abordable.

Les chercheurs ont déjà mis au point divers systèmes intéressants, mais ils se sont rendu compte que, en augmentant le nombre de locuteurs dont l'ordinateur peut reconnaître la voix, le nombre de mots qu'il peut identifier diminue en proportion. En moyenne, un système de ce genre peut reconnaître entre vingt et trente mots à la fois, avec un taux de réussite de 85 à 90 %.

L'utilisation potentielle de tels systèmes est très vaste. Les postes allemandes en utilisent un pour le tri du courrier; l'aéronautique constitue aussi un champ d'application prometteur, dans le domaine civil comme dans le domaine militaire : les pilotes sont surchargés de tâches pendant le vol. Dans de telles situations, le nombre de mots reconnaissables à la fois est d'une vingtaine. Il y a pourtant un moyen d'accroître le rendement. L'utilisateur sélectionne l'un des vingt mots à partir d'un menu; chaque commande identifiée permet le passage à un nouveau menu, composé de nouveaux mots entre lesquels on pourra faire un choix. L'ordinateur n'entreprendra une action quelconque qu'après reconnaissance d'une séquence entière. Dans le cas du tri postal, le premier niveau de tri se fait par départements, puis par villages,

Les éléments de la parole

Une des techniques de reconnaissance de la parole se contente de numériser le signal émis, et de se livrer à de vastes recherches de « reconnaissance de formes ». Une méthode plus efficace consiste à utiliser un prétraitement du signal : des circuits indépendants permettent de mesurer les éléments « sonores » (comme les voyelles), les fricatives (s, f, z, etc.) et les brèves périodes de silence (entre deux syllabes, par exemple). (Cl. Kevin Jones.)



etc. Ce n'est qu'une fois l'opération entièrement achevée que la lettre ou le colis seront expédiés, ce qui assure une fiabilité maximale.

L'analyse de la voix

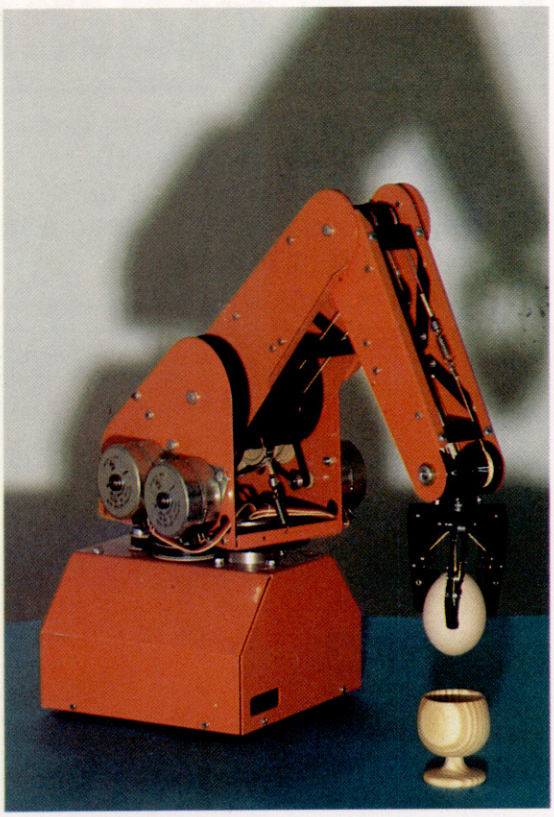
La reconnaissance de la parole peut être accomplie de deux façons différentes.

La première (la plus « rapide ») consiste simplement à faire passer le message par un convertisseur analogique-numérique, l'ordinateur accomplissant toute l'analyse. Mais cette méthode a plusieurs inconvénients, en particulier de prendre beaucoup de temps. De tels systèmes ont parfois besoin de deux à trois secondes pour identifier un message donné. Or, pour être réellement utiles, il leur faudrait « comprendre » la parole aussi vite qu'un être humain; ce que les calculs ne permettent pas.

La seconde méthode utilise le prétraitement. Plutôt que d'analyser mathématiquement le signal, on se contente de faire usage des procédés électroniques classiques. L'ordinateur est soumis à un ensemble d'informations (la fréquence, l'énergie, la hauteur, etc. du message prononcé). Les fréquences sont par exemple calculées en filtrant le signal, en relevant le niveau de chaque fréquence particulière, un peu comme on se sert d'un égaliseur sur une chaîne haute fidélité, pour « faire sortir » la batterie. Ce type d'analyse électronique s'effectue au moment même de la réception du signal; il est donc pratiquement instantané. La même opération, accomplie à partir des données numériques fournies par un convertisseur analogique-numérique, nécessiterait l'emploi simultané de plusieurs ordinateurs. La méthode par prétraitement, si elle en est encore au stade de la recherche (il n'existe aucun dispositif de ce genre qui soit commercialisé), offre sans doute les possibilités les plus riches.

Une fois rassemblées les informations relatives à la fréquence, à la hauteur, etc. (et quel que soit le procédé choisi), la reconnaissance est accomplie par comparaison des renseignements obtenus avec des modèles stockés dans la mémoire de l'ordinateur. Les mots à identifier sont d'abord présentés un par un au système, et les résultats sont stockés dans une « bibliothèque » numérique d'exemples. Puis toute la phrase est de nouveau présentée à l'ordinateur, et le tout est comparé aux modèles de base. Si la confrontation est concluante, ce second choix d'exemples est ajouté au premier, ce qui permet d'obtenir une version plus complète. Il peut s'agir d'un processus continu, au cours duquel de nouveaux locuteurs fournissent de nouvelles informations.

Pour reconnaître un mot prononcé par quelqu'un, l'ordinateur doit comparer l'information qu'il contient avec un ou plusieurs des modèles qu'il abrite en mémoire. Dans de nombreux cas, il y aura plusieurs possibilités. Les deux premières syllabes d'*international* sont par exemple les mêmes que celles d'*interprète*. C'est par un ensemble de comparaisons successives que l'ordinateur peut identifier un mot qui lui a été présenté.



Contrôle de l'environnement
Les applications les plus récentes des systèmes de reconnaissance de parole sont généralement de nature éducative. L'une d'elles est appelée « environnement limité »; elle met en œuvre un ordinateur, un bras articulé et un nombre limité d'objets simples, que le bras peut manipuler. Parlant dans un micro, l'utilisateur peut ainsi demander au bras de PLACER L'ŒUF DANS LE COQUETIER. Le rôle de l'ordinateur est d'interpréter les instructions, puis de rechercher dans sa mémoire les positions respectives des objets. (Cl. Ian McKinnell.)

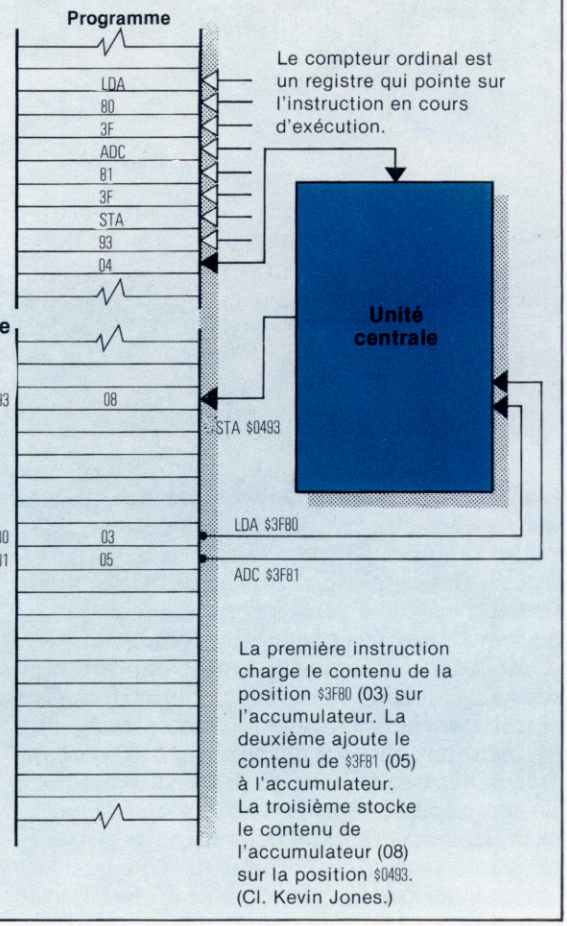
Les systèmes de reconnaissance de la parole trouveront certainement de nombreuses applications; mais leur principal emploi interviendra au sein de logiciels complexes, comme les bases de données, où les commandes sont sélectionnées à partir d'un menu affiché sur écran. Cela permettrait de surmonter le pire obstacle pour les non-spécialistes : le clavier. Les systèmes vidéotex comme Télétel en France l'ont réduit à un simple ensemble de touches alphanumériques, mais cela limite sérieusement les possibilités d'intervention de l'utilisateur. Une interface commandée par la voix, capable d'identifier un ensemble standard de commandes d'interrogation de données, ainsi que des symboles numériques et les lettres de l'alphabet, serait un outil de travail très puissant.

Les systèmes de reconnaissance de parole actuellement disponibles sur le marché restent toutefois très primitifs, même s'il suffit de les brancher sur l'ordinateur. Big Ears ou le Speech Lab de Heuristic Inc. ne peuvent identifier plus de quelques mots, prononcés par une seule personne, en dépit de leur puissance de traitement. Avant d'être d'usage général, de tels appareils devront d'abord pouvoir reconnaître des mots émis par n'importe qui, quels que soient les problèmes d'accent ou de dialecte. Le principal obstacle est actuellement celui de la capacité mémoire nécessaire au stockage des modèles. On a d'ailleurs pensé à utiliser pour cela un disque vidéo, qui aurait le double avantage de ne consommer aucune mémoire interne et de n'entraîner qu'une faible perte de temps.

Code machine

Apprendre à programmer en code machine représente un saut considérable par rapport à la programmation en basic, avec à la clé, efficacité et rapidité.

Étape par étape



Jusqu'à présent, *ABC Informatique* n'a abordé que la programmation en BASIC. Cela s'explique par l'intérêt du BASIC, qui est d'un usage général et très accessible. Cependant, avec l'expérience, vous entreprendrez des travaux de programmation de plus en plus complexes et découvrirez les limites du BASIC. Le déplacement à l'écran des graphiques vous semblera long, et vous réaliserez qu'il vous faut trop souvent avoir recours aux commandes délicates PEEK et POKE.

Par contraste, la programmation en code machine impose peu de contraintes et semble très rapide. Néanmoins, malgré cet intérêt, peu de programmeurs amateurs franchissent ce pas. Cela est dû en partie au travail astreignant et à la différence radicale d'approche que cette programmation suppose. Il convient au moins de comprendre ce qu'est le code machine. Cet article, le premier de deux, expose les principales procédures.

Le code machine, ainsi que nous l'avons vu, constitue le langage du microprocesseur. Celui-ci est le cœur de l'unité centrale, elle-même base de votre micro-ordinateur. Le processeur ne peut exécuter que des fonctions très simples (ajouter deux chiffres dans un nombre, mais pas les multiplier). Cela dit, il exécute ces fonctions élémentaires à une vitesse très élevée. Chaque tâche du microprocesseur est spécifiée en termes de cycles d'horloge, durée nécessaire à son accomplissement. Si votre unité centrale fonctionne à 1 MHz, un cycle d'horloge est de 1 microseconde, et une tâche comprenant quatre cycles s'effectue en 4 millionièmes de seconde.

En conséquence, tout programme écrit en code machine sera composé de très nombreuses instructions, et chaque fonction devra être décomposée en opérations élémentaires. La programmation en code machine consistera à manipuler les bits ou les octets de mémoire avec des opérations arithmétiques binaires très simples, et des fonctions logiques de base telles que ET, OU et NON.

C'est une des raisons pour lesquelles l'écriture en code machine est lente. L'autre raison est que le programmeur doit savoir exactement comment est disposée la mémoire. En BASIC, une instruction telle que LET A = 5 confie à l'interpréteur le soin de trouver un emplacement mémoire pour la variable. En outre, il saura où chercher les données lorsque A sera ultérieurement évoqué dans le programme. Quand vous commencez à programmer en code machine, vous découvrez qu'il faut donner une adresse (emplacement mémoire) pour chaque donnée à stocker. Il faudra en outre vous assurer que cette adresse ne sera pas effacée par l'écriture d'une autre donnée.

Voyons en quoi consiste le code machine. Nos exemples se réfèrent à des unités centrales de 8 bits, telles que les Z80 et 6502. Les unités centrales à 16 bits fonctionnent de la même manière, mais traitent à chaque opération deux fois plus de bits.

Le microprocesseur est connecté à la mémoire de l'ordinateur par deux connexions ou BUS (un groupe de câblages) : le BUS d'adresse et le BUS de données. Il existe également un BUS de contrôle, mais il est destiné à fournir les signaux de synchronisation de l'unité centrale, et n'est pas utilisé par le programmeur.

Le BUS d'adresse est sur 16 bits. En disposant les bits d'une certaine manière, l'unité centrale choisit l'un des 65 536 octets de sa carte mémoire. Pour un micro, ces emplacements sont destinés soit à la RAM, soit à la ROM, soit aux composants spécifiques d'entrée-sortie, d'autres étant inutilisés. Quand l'unité centrale veut lire une position mémoire (une des lignes du BUS de contrôle indique s'il s'agit de lire ou d'écrire), l'octet choisi transmettra sa configu-

ration au BUS de données, sous la forme d'une disposition de 8 bits. Inversement, l'unité centrale peut inscrire une configuration de 8 bits sur une position. L'unité centrale ne sait pas quelle partie de la mémoire est allouée à la RAM ou à la ROM, aussi l'adressage est-il sous la responsabilité du programmeur.

Le microprocesseur comporte peut-être une douzaine de registres, ou emplacements mémoire, utilisés pour stocker les résultats temporaires, et pour effectuer les fonctions logiques et arithmétiques (binaires). La plupart de ces registres portent sur 1 octet mémoire, bien que certains portent sur 2 octets.

Un de ceux-ci est appelé « compteur ordinal » (registre), et comporte l'adresse mémoire de l'instruction en code machine en cours d'exécution. Cette adresse mémoire est assimilable au numéro de ligne d'un programme en BASIC.

Un autre type de registre, le registre « accumulateur » (sur 8 bits seulement), est le totalisateur. Des octets lui sont ajoutés ou retirés; il est destiné aux opérations arithmétiques. Pour résumer, nous pouvons spécifier un exemple de code machine de la façon suivante :

1° Charge l'accumulateur du contenu de la position mémoire $3F80. Les adresses en code machine sont généralement en numération hexadécimale. Les nombres hexadécimaux sont indiqués par un préfixe, habituellement un $.

2° Ajoute à l'accumulateur le contenu de la position mémoire $3F81; le résultat peut être supérieur à ce qu'un seul octet peut contenir, auquel cas il y aurait également un bit de retenue.

3° Stocke le nouveau total de l'accumulateur sur la position mémoire $0493.

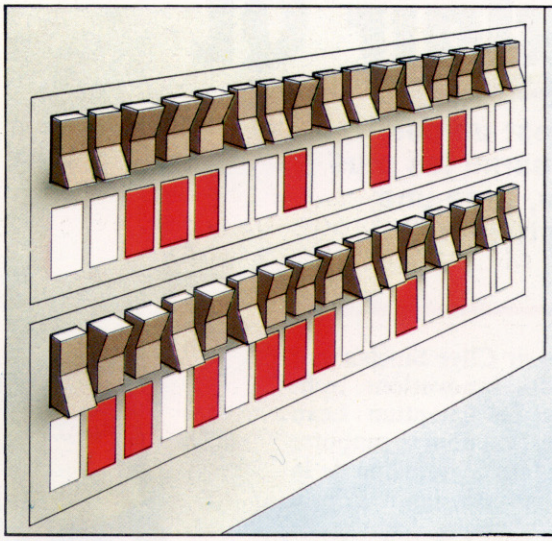
Chacune de ces opérations est une instruction en code machine. Le programme s'écrira ainsi :

```
LDA $3F80 (Load Accumulator)    [charge l'accumulateur]
ADC $3F81 (ADd with Carry)      [ajoute avec retenue]
STA $0493 (STore Accumulator)   [stocke sur l'accumulateur]
```

Les commentaires entre parenthèses, semblables aux instructions REMark BASIC, sont sans effet. Le premier enregistrement de chaque ligne s'appelle le « code opération »; il indique la nature du traitement. La deuxième colonne contient l' « opérande », les détails, le contexte, les données concernées. Un microprocesseur est généralement susceptible d'effectuer plusieurs douzaines d'opérations simples, et chaque code opération occupera seulement 1 octet en mémoire, une fois entré.

Un code opération peut donc être compris entre 0 et 255 (ou plus exactement une valeur hexadécimale comprise entre $00 et $FF). Il est cependant plus commode, lors du développement d'un programme, d'utiliser pour rendre le listage plus lisible des lettres significatives telles que LDA, ADC ou STA.

Chacun des opérandes indiqués ici est un nombre sous la forme d'une valeur hexadécimale comprise entre $0000 et $FFFF et porte sur 2 octets. Cependant, certains opérandes ne sont que sur un seul octet, et certains codes opérations ne comportent pas d'opérande. Le court programme donné plus haut prendra seulement 9 octets, sans compter les trois positions mémoire qui contiennent le programme ($3F80, $3F81, et $0493).



Clignotants
Le cliché représentant au cinéma les ordinateurs munis de panneaux énormes de clignotants provient du panneau frontal de nombreux mini-ordinateurs. Ce dernier comportait des rangées de clignotants et d'interrupteurs correspondants aux adresses de l'unité centrale et aux bus de données ou connections de données. Auparavant, les programmes en code machine devaient être entrés en binaire sous cette forme.
(Cl. Kevin Jones.)

Pour cet exercice facile à exécuter, le programme écrit en BASIC donnerait le même résultat, mais occuperait 50 octets. En outre, et plus délicat, il exécuterait les opérations une centaine de fois plus lentement, étant donné le temps que doit prendre l'interpréteur pour transcrire le programme.

```
10 A = PEEK (16256)
20 A = A + PEEK (16257)
30 POKE 1171,A
```

N.B. — Les emplacements mémoire utilisés par ce programme ne conviennent pas nécessairement à votre machine.

Dans un prochain numéro d'*ABC Informatique*, nous verrons comment le code machine est saisi et exécuté sur un micro-ordinateur personnel, et les différentes expressions que le code machine peut prendre.

LDA-LoaD Accumulator
(charge l'accumulateur) transfère sur le registre accumulateur interne le contenu d'une position mémoire (1 octet).

STA-STore Accumulator
(stocke l'accumulateur) vide l'accumulateur sur une position mémoire (l'inverse de LDA).

ADC-ADd with Carry
(ajoute avec retenue) ajoute le contenu d'une position mémoire au contenu courant de l'accumulateur, créant éventuellement une retenue.

SBC-SuBstract with Carry
(soustrait avec retenue) est la fonction inverse de ADC.

JMP-JuMP
(saut d'adresse) transfère le traitement en cours sur une autre position mémoire : semblable à une instruction GOTO en BASIC.

Codes opérations
Voici quelques codes opérations (type de traitement ou instruction) qu'un microprocesseur exécute.

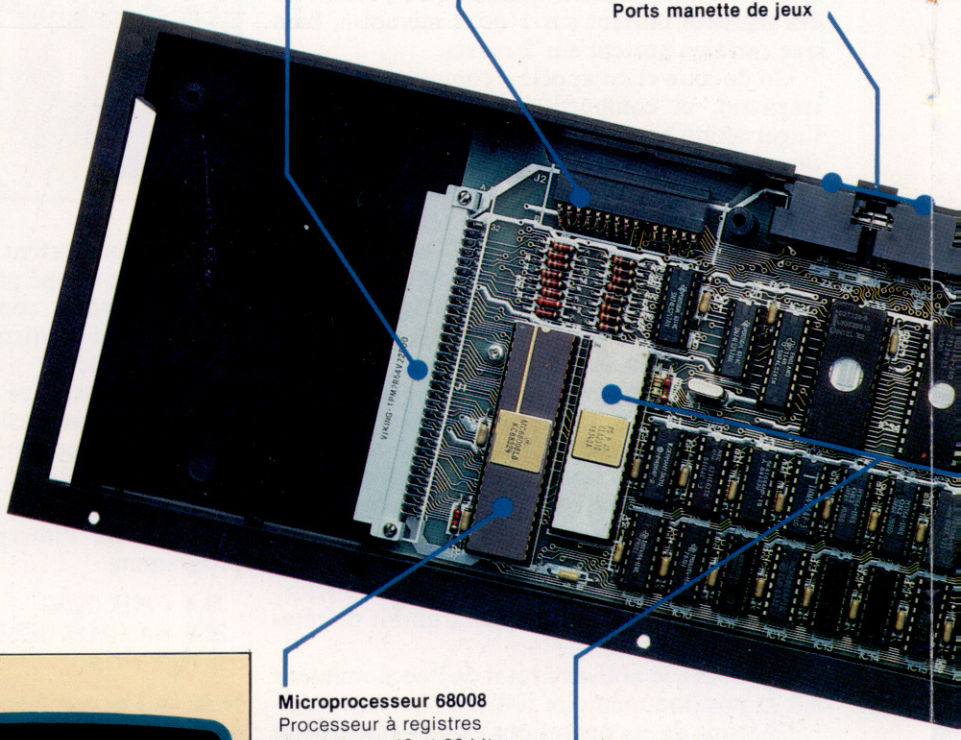
Le Sinclair QL

Le Quantum Leap est un des microprocesseurs parmi les plus avancés du marché, avec une mémoire pouvant atteindre 0,5 méga-octet.

Les précédents micros de sir Clive Sinclair ont certainement constitué des innovations techniques et commerciales. Le Quantum Leap (QL) est le dernier en date. Son nom se rapporte à un « bond en avant » dans le domaine de la micro-informatique. Il s'adresse, pour un prix raisonnable, soit aux amateurs éclairés de micro-informatique, soit aux utilisateurs professionnels. Les applications domestiques lui sont tout naturellement destinées. Il est donc en concurrence directe avec des micros comme le Commodore 64, mais en termes de spécifications techniques il semble qu'il leur soit absolument supérieur.

Il est évident que le QL a été conçu pour réunir toutes les techniques d'avant-garde. Tranchant sur l'alternative traditionnelle pour le choix du microprocesseur, Z80 ou 6502, l'unité centrale est bâtie autour d'un microprocesseur de la famille du Motorola 68000. Ce dernier est à l'heure actuelle le plus performant des microprocesseurs et il est utilisé sur des machines telles que le Lisa d'Apple. Il s'agit du Motorola 68008 ; son bus externe de données est donc sur 8 bits. Bien que ses registres internes soient sur 16 bits et qu'il puisse mener à bien des fonctions sur 32 bits, le chargement et le stockage sur registres devront se faire par moitié. Le fonctionnement de l'unité centrale s'en trouvera donc ralenti. En contrepartie, le prix de revient des composants mémoire reste très bas, ce qui constitue une des préoccupations majeures de Sinclair.

Le QL est livré avec 128 K de RAM en standard, mais peut atteindre une capacité mémoire maximale de 512 K (0,5 méga-octet) par l'adjonction de cartes mémoire. Cette mémoire importante est particulièrement utile pour les applications professionnelles de gestion, dans la mesure où elle permet de réduire la fréquence des appels du programme à des unités de mémoire externes. Le stockage est ici interne et



Interface extension
Couplage possible des périphériques et jusqu'à 0,5 méga-octet de RAM additionnelle.

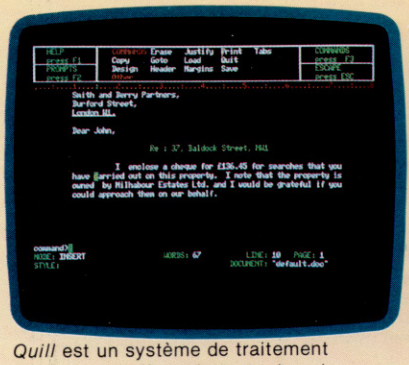
Cartouche ROM
Enfichage possible de 32 K de ROM supplémentaire.

Ports manette de jeux

Microprocesseur 68008
Processeur à registres internes sur 16 et 32 bits, avec bus externe de données sur 8 bits.

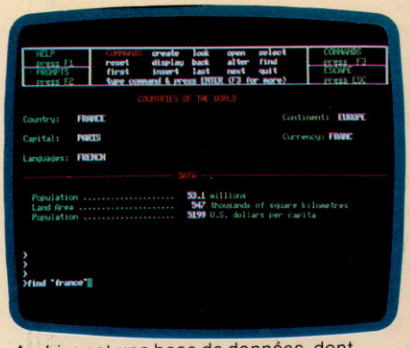
Composants sur mesure
Composants propres à l'ordinateur, ici au nombre de deux, pour l'affichage et pour diverses interfaces.

Logiciel pour QL

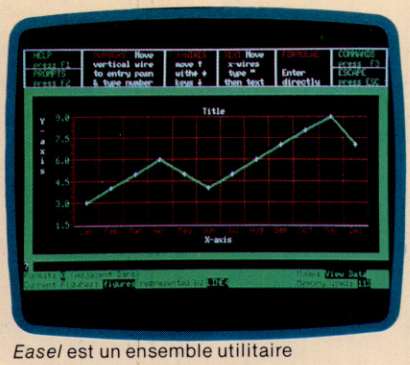


Quill est un système de traitement de texte qui affiche le texte dans le format même d'impression.

Abacus est un tableur qui réfère les positions par des noms au lieu de coordonnées.



Archive est une base de données, dont les présentations d'enregistrements sont déterminées par l'utilisateur.



Easel est un ensemble utilitaire graphique créant des courbes et des tableaux. (Ian McKinnell.)

Clavier

Bien que conçu sur un modèle à membrane (protection accrue), il présente soixante-cinq touches à détente complète, et son toucher vaut bien celui des machines les plus élaborées de gestion. Il comporte quatre touches à curseur, et cinq programmables. Le signe copyright (©) y figure aussi.

SINCLAIR QL

PRIX

**

TAILLE

472 × 138 × 46 mm.

UC

Motorola 68008.

VITESSE DE L'HORLOGE

7,5 MHz.

MÉMOIRE

128 K de RAM, ext. à 512 K; 32 K de ROM ext. à 64 K.

AFFICHAGE VIDÉO

25 lignes de 85 caractères (avec moniteur), graphique haute résolution : 512 × 256 pixels (4 couleurs), 256 × 256 pixels (8 couleurs).

INTERFACES

2 RS232, 2 manettes de jeu, microlecteurs; LAN, TV, RVB (moniteur).

LANGAGE INTÉGRÉ

BASIC.

AUTRES LANGAGES DISPONIBLES

Plusieurs langages prévus, notamment le langage C.

ACCESSOIRES FOURNIS

Manuel utilisateur, 4 programmes d'applications.

DOCUMENTATION

Manuel provisoire.

Ports série

Deux ports RS232 pour la gestion d'une imprimante et d'un modem. L'interface la plus répandue d'imprimante (Centronics) doit être acquise en supplément.

Prise téléviseur

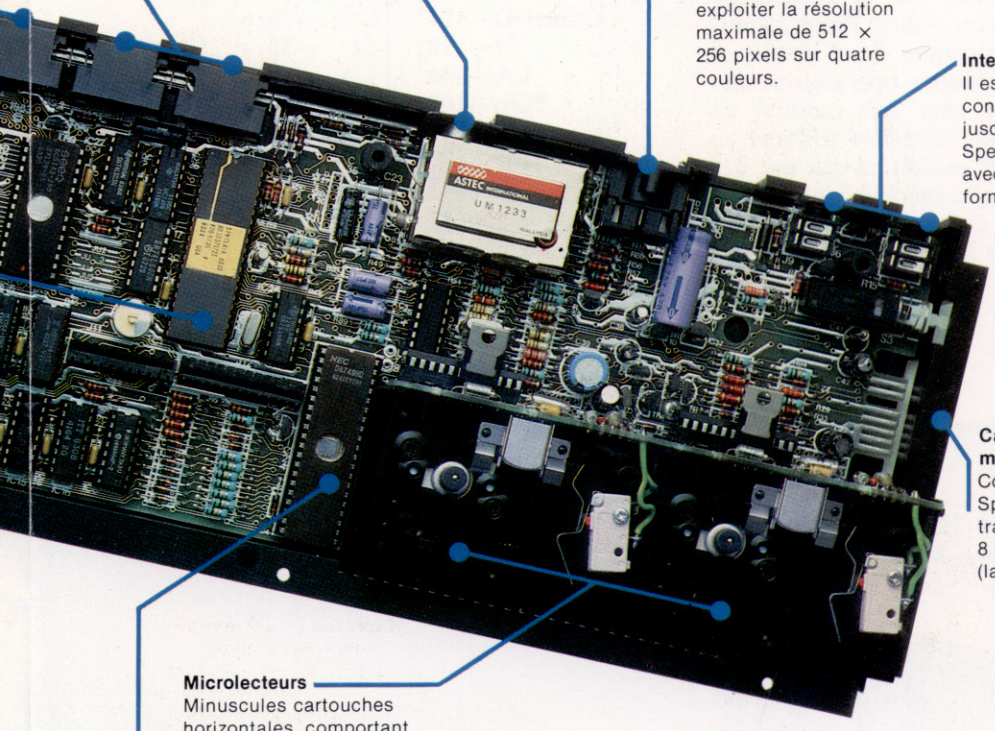
Le QL ne peut afficher sur un téléviseur que 60 ou 40 colonnes, contre 85 avec un moniteur.

Port moniteur

Contrairement au Spectrum, le QL peut gérer directement un moniteur rouge-vert-bleu, précieux pour exploiter la résolution maximale de 512 × 256 pixels sur quatre couleurs.

Interface réseau

Il est possible de connecter ensemble jusqu'à 64 micros QL et Spectrum (ce dernier avec l'interface 1), pour former un réseau local.



Cartouche extension microlecteurs

Comme pour le Spectrum, le QL peut traiter jusqu'à 8 microlecteurs. (Ian McKinnell.)

Microlecteurs

Minuscules cartouches horizontales, comportant chacune une bande sans fin, pouvant stocker 100 K chacune.

Deuxième microprocesseur

De type Intel 8049, contrôle le clavier, le son et les ports série. Il laisse au 68008 le soin de gérer les programmes utilisateur.

consiste en deux microlecteurs incorporés d'une capacité mémoire de 100 K chacun. Bien que le QL soit complètement autonome, les microlecteurs apparaissent comme un point faible en comparaison de l'efficacité du processeur.

L'argument de Sinclair est qu'il est prévu une interface Winchester (disque dur); mais il n'en reste pas moins que rien n'est prévu pour les lecteurs de disquettes (des fabricants indépendants ne manquent pas d'en proposer rapidement). Sans disquettes, le QL ne pourra malheureusement pas fonctionner sous le système d'exploitation UNIX. Ce dernier est généralement une des raisons majeures du choix du microprocesseur Motorola 68000, et peut être considéré comme le remplaçant probable du CP/M en tant que système d'exploitation supportant des logiciels de gestion.

Le QL est livré en standard avec quatre progiciels de gestion. Quill est un traitement de texte; Abacus, un tableur; Archive, une base de données; Easel, un progiciel graphique. Ils fonctionnent tous sous le système d'exploitation résident, baptisé QDOS. Le succès que cette machine risque de connaître incitera sûrement de nombreux auteurs de logiciels à créer des programmes pour elle, bien qu'il ne leur soit pas facile de transcrire leurs programmes déjà existants. Cela dit, les micros Sinclair visent avant tout à jouer le rôle de pionnier sur le marché. Le BASIC résident est une version améliorée du BASIC du Spectrum, et s'appelle maintenant en toute modestie SuperBASIC! Il comporte des commandes pour traiter les procédures (ce qui encourage la programmation structurée) et pour avoir accès au système d'exploitation depuis un programme BASIC. Le BASIC et le QDOS figurent tous les deux dans les 32 K de mémoire morte en standard (ROM).

Le QL de Sinclair est incontestablement une machine intéressante. Mais, plus important, c'est une machine qui est conçue pour accueillir tellement d'extensions qu'elle n'est pas prête d'être dépassée. C'est une étape importante dans la lignée des Sinclair : ZX-80, ZX-81 et Spectrum.

CARTE PROTOTYPE

Cette photographie représente les cartes des circuits de présérie du QL. Les éléments décrits peuvent donc être modifiés.

Principes sonores

Les fonctions sonores des modèles Atari comportent quatre voix indépendantes.

Les capacités sonores Atari sont bonnes — on peut s'en rendre compte avec les jeux en cartouche — même si leur contrôle est quelque peu fantaisiste. Quatre oscillateurs indépendants d'ondes sont prévus, chacun sur trois octaves. En outre, la sortie oscillateur peut subir une distorsion selon sept modes différents pour moduler le son. Ces capacités sonores sont accessibles par les commandes BASIC *via* la commande SOUND. Le composant son Atari, Pokey, comporte d'autres caractéristiques dont les filtres pour les aigus, et des modes opératoires spéciaux. Le son produit peut être modifié de manière très variée. Les capacités sonores ne peuvent être pleinement exploitées que par des commandes POKE ou code machine. Mais cela dépasse le cadre de ce cours. La sortie du son passe exclusivement par le haut-parleur du téléviseur.

SON

La commande est très simple au format suivant :

SOUND O,P,D,V
O = Oscillateur (0-3)
P = Pitch (intensité) (0-255)
D = Distorsion (nombres entiers 0-14)
V = Volume (1-15)

Chaque commande SOUND ne peut choisir qu'un seul oscillateur ; aussi est-il impossible de mettre en œuvre plusieurs oscillateurs à la fois. En présence d'une partition faisant intervenir les quatre oscillateurs, le retard des réponses sera relativement important.

Pitch est calculé de manière un peu bizarre et certaines fréquences sont imprécises. Les fréquences décroissent selon l'accroissement de l'intensité (Pitch), avec une gamme effective allant du *do* à 29 (1046,5 Hz) jusqu'au *do* à 243 (130,81 Hz). La table suivante donne les valeurs d'intensité pour les notes.

Octave-1	Octave-3
(Mid)DO—121	DO —29
SI—128	SI —31
LA—144	LA —35
SOL—162	SOL—40
FA—182	FA —45
MI—182	MI —47
RE—217	RE —53
DO—243	DO —60

Le paramètre de distorsion « P » est équivalent au canal bruit sur la plupart des ordinateurs, mais avec davantage de ressources. Chaque nombre entier suscite une combinaison différente d'impulsions aléatoires s'ajoutant à la sor-

Éclat de lumière

Un aperçu rapide du graphisme Oric révèle de nombreuses similitudes avec le Spectrum.

Le micro-ordinateur Oric-1, commercialisé courant 1983, est manifestement destiné à rivaliser directement avec le ZX Spectrum de Sinclair. Oric permet quatre modes d'affichage. Un seul mode, cependant, autorise l'affichage haute résolution. Huit couleurs sont possibles ; les couleurs de fond et d'affichage étant respectivement attribuées par les commandes INK et PAPER.

Le BASIC d'Oric comporte plusieurs commandes haute résolution destinées à venir en aide au programmeur d'applications graphiques.

L'écran contient 28 lignes de 40 caractères. Ces caractères ne reposent pas, comme d'habitude, sur une grille de 8 par 8 pixels, mais de 8 par 6. En mode haute résolution l'écran est de 240 × 200 pixels ; les trois lignes inférieures étant réservées pour des informations du type « message d'erreur ». Il n'existe pas de commande du type PAINT, mais avec un peu d'imagination et à l'aide de la commande FILL, on peut très bien la mettre en œuvre. Comme pour le Spectrum, il est possible de mélanger affichage haute résolution et texte sur le même écran. La différence étant qu'Oric permet d'attribuer une couleur à chaque ligne d'une grille. Le Spectrum, quant à lui, n'autorisait qu'une seule couleur par grille.

Voyons maintenant les modes d'affichage basse résolution de l'Oric-1. Ils sont au nombre de trois : TEXT, LORES0 et LORES1, les deux derniers ne différant que par leur jeu de caractères. Le

tie standard de l'oscillateur. Curieusement, 10 produit un signal sans distorsion qui n'est pas 0. Avec un peu d'entraînement, les sons de distorsion produisent des effets sonores intéressants.

Le volume V peut recevoir une valeur comprise entre 0 et 15; une moyenne raisonnable étant 7 ou 8. Vous remarquerez qu'il n'existe pas de moyen commode pour contrôler la durée d'une note ou les pauses entre les notes. La solution est d'utiliser des boucles soigneusement synchronisées FOR...TO...NEXT.

Pour illustrer l'utilisation de SOUND, les commandes suivantes joueront un *sol* dans la troisième octave, sur l'oscillateur 1, à un volume de 8, pour 50 boucles FOR...TO...NEXT.

```
10 SOUND 1,40,10,8
20 FOR N = 1 TO 50:NEXT N
30 END
```

END en ligne 30 éteint l'oscillateur. Il est également possible de faire intervenir une nouvelle commande SOUND pour le même oscillateur : elle arrêtera la note pour en jouer une autre. Un programme destiné à jouer un air très simple pourra prendre l'aspect suivant :

```
10 REM*UN P'TIT AIR...*
20 FOR I=1 TO 7
30 READ N:REM *NOTE*
40 SOUND 3,N,10,7:REM *JOUER UNE NOTE*
50 FOR P=1 TO 400:NEXT P:REM *PAUSE*
60 NEXT I
70 DATA 219,162,128,144:REM *RÉ SOL SI LA*
80 DATA 162,193,162:REM *SOL MI SOL*
90 END
```

Les ressources du composant Pokey d'Atari sont exploitables à partir de la commande BASIC POKE appliquée aux positions mémoire 53760 et 53763. Les routines de son peuvent ainsi s'exécuter plus rapidement, et tous les oscillateurs peuvent être mis en œuvre en même temps. Tout ce que vous pouvez savoir à ce sujet, ainsi que des techniques plus hasardeuses de code machine figurent dans le livre *De Re Atari* publié par Atari.

mode TEXT permet le positionnement horizontal par la commande TAB. Les deux modes LORES autorisent en outre le positionnement vertical par la commande PLOTx,y,A$ où x et y sont les coordonnées de la position, et A$ la chaîne (mot ou phrase) à afficher par PRINT. Le petit programme suivant utilise cette commande pour afficher verticalement un mot.

```
10 REM AFFICHAGE VERTICAL D'UN MOT
20 CLS
30 LORES0
40 A$ = « JULES »
50 FOR X = 1 TO 5
60 B$ = MID$ (A$, X, 1)
70 PLOT 16, 11 + X, B$
80 NEXT I
90 END
```

La commande HIRES permet de passer en mode haute résolution. L'écran a alors son origine dans le coin supérieur gauche.

Le BASIC Oric comporte plusieurs commandes spécifiques graphiques. CURSETx,y,k positionne le curseur sur le point de coordonnées (x, y). La valeur de « k » permet d'utiliser conjointement certaines fonctions.

Valeurs de k	Fonction
0	tracé en couleur d'arrière-plan
1	tracé en couleur de premier plan
2	inversion des couleurs
3	sans effet

CURMOVx,y,z est semblable à CURSET à la différence près que le déplacement du curseur dépend de la position antérieure. DRAW trace une ligne droite depuis la position courante du curseur jusqu'au point de positions x horizontalement et y verticalement. CIRCLEr,k trace un cercle de rayon r. PATTERNn constitue une commande tout à fait originale. Elle divise les tracés de lignes ou de cercles en une suite de points et de traits. Les dispositions exactes sont définies par n qui est compris entre 0 et 255. Oric utilise la séquence binaire de n pour réaliser ces pointillés. Voici un exemple :

Valeur de n	Équivalent binaire	Disposition
170	10101010	- - - - - - - -
15	00001111	_ _

En dernier lieu, existe la commande FILLa,b,n. Chaque suite de positions, concernant un caractère à l'écran, est associée à un nombre qui informe sur les couleurs d'arrière- et de premier plan, sur le caractère présent et son état. Ce nombre est appelé « attribut » de cette suite de positions. FILLa,b,n remplit b positions sur a suites, avec les attributs représentés par « n ».

```
10 REM CONE
20 HIRES
30 CURSET 120,0,3
40 PAPER 3:INK 4
50 FOR R=1 TO 65
60 PATTERN 200-R
70 CURMOV 0,2,3
80 CIRCLE R,1
90 NEXT R
100 END
```

PATTERN pour un cône
Ce programme met en œuvre les ressources haute résolution de l'Oric-1. Un cône est dessiné en traçant des cercles de rayon croissant. Remarquez également l'utilisation de PATTERN pour obtenir un effet de pointillé.

Cryptographie

La cryptographie a été l'une des premières applications de l'ordinateur. De nos jours, le chiffrement et le déchiffrement d'une donnée sont à la portée du programmeur basic.

Toute communication est codifiée, que le langage soit parlé ou écrit. L'intelligibilité du code dépend de la cible. La parole est destinée à être comprise au moins par celui ou celle à qui elle s'adresse. Les dialogues avec l'ordinateur relèvent de la même démarche. Les ordinateurs personnels sont ouverts à une forme de langage bien particulière, le BASIC, qui a néanmoins le don d'être accessible au plus grand nombre possible d'utilisateurs. Mais nous savons que la machine, en elle-même, n'utilise pas ce langage pour exécuter ses fonctions. Les énoncés en BASIC doivent d'abord être interprétés sous une forme purement numérique, puis être utilisés pour mettre en place les séquences de commutation définies dans le programme. Ces formes de codes sont d'un accès facile et délibérément non restreint.

Compression des données

Les utilisateurs voulant stocker de nombreux fichiers cherchent à comprimer les données pour optimiser la place disponible. Un des moyens est de représenter une chaîne par un seul caractère. De même que le ZX Spectrum code un mot entier « réservé BASIC » sur une seule touche, un symbole peut venir remplacer un mot ou même une phrase d'utilisation très courante. Un utilitaire compact comme Unix est connu pour compresser les données de 38 % ; Clip, fonctionnant sous CP/M, donne encore de meilleurs résultats. Compactor, pour le Commodore 64, poursuit la même finalité pour les programmes BASIC en supprimant les REM, les espaces superflus, etc.

Il existe un autre type de codage (appelé chiffrement) qui, lui, s'oppose radicalement à la communication la plus simple. Sa raison d'être est précisément de limiter la diffusion d'une information. Jusqu'à la seconde moitié du XX^{e} siècle, le chiffrement des informations était réservé aux gouvernements et à quelques gros intérêts industriels. Le développement du téléphone et sa grande utilisation, même pour des informations confidentielles de type commercial, ont rendu la communication au sens large plus vulnérable, et ont permis le développement du codage des informations à l'aide des ordinateurs.

Le chiffrement et le codage partent de techniques simples — ajouter ou soustraire à tous les octets une même valeur, par exemple, ou substituer un caractère à un autre — pour s'étendre aux techniques les plus complexes qui suivent de près des théories très avancées sur les nombres mathématiques. Ces « chiffres » ne comportent aucun élément de répétition et ne sont donc pas vulnérables aux méthodes de décodage par analyse de fréquence.

La plus simple des techniques de codage est la méthode dite « chiffre de Jules César », qui date des Romains. Elle ne demande que le message et la clef. En voici un exemple :

```
AMSPQ B' GLDMPKYRGOSC
NPYRGOSC CR DYKGJGYJC
```

Nous pouvons émettre des suppositions sur les mots codés, en fonction des espaces les séparant (bien qu'ils puissent être également trompeurs). Ce que l'on peut dire du message c'est qu'il comporte cinq mots ayant respectivement 5, 12, 8, 2 et 9 lettres. Nous pouvons également supposer que les deuxième, troisième et cinquième mots se terminent par la même lettre. Celle-ci (C) est l'une des trois lettres du message qui apparaît avec la fréquence la plus élevée (les deux autres étant G et Y). Cette constatation est de la plus haute importance pour celui qui décrypte le message, pour autant qu'il sache de quelle langue il s'agit. En français, la lettre la plus fréquente est E, en particulier à la terminaison des mots.

Sur un échantillon aussi réduit (37 lettres), incertain pour une analyse significative de la fréquence des lettres, les résultats du décryptage sont douteux. Mais essayons tout de même la substitution des lettres selon leur fréquence, et d'abord remplaçons C par E :

```
AMSPQ B' GLDMPKYRGOSe NPYRGOSe
eR DYKGJGYJe
```

Le message est toujours aussi incompréhensible, mais il existe d'autres clefs. Ainsi l'éloignement dans l'alphabet de la lettre d'origine de celle qui lui est substituée. C est placé deux lettres avant E; essayons d'appliquer ce principe aux autres lettres : deux positions après G et Y (les deux autres lettres très fréquentes), nous trouvons I et A :

```
AMSPQ B' iLDMPKaRiOSe NPaRiOSe eR
DaKiJiaJe
```

Le dernier mot se trouve maintenant composé de la manière suivante : «[quelque chose] — voyelle — [quelque chose] — voyelle — [quelque chose] — voyelle — [quelque chose] — voyelle », ce qui représente une construction très vraisemblable en français. En outre, la dernière lettre est E, ce qui est souvent le cas. Nous sommes donc sur une piste plausible. Continuons avec le reste des lettres du premier mot, et

nous obtenons : COURS D'INFORMATIQUE...

Le chiffre de César est donc un code qui consiste à faire « glisser » les lettres de l'alphabet plusieurs positions en deçà ou au-delà de la leur. Il peut être affiné en appliquant une chaîne de valeurs de déplacement à la suite des lettres : la première lettre étant décalée de deux valeurs en deçà (par exemple); la deuxième lettre, de trois, etc. Ce qui se note ici — 234... Lorsque la dernière valeur de décalement a été appliquée, on repasse à la première pour la lettre suivante.

Le chiffre de César

Ce programme (il est écrit en BASIC Commodore) code selon le chiffre de César un texte avec une chaîne de valeurs de déplacement à cinq codes. Le message apparaît en clair, et, lorsque le retour chariot est tapé, la version codée est affichée. Le message doit être tapé sans espace et sans ponctuation.

```
10 INPUT «ENTER A FIVE FIGURE KEY»;K$        (clef à cinq codes)
20 INPUT «ENTER THE MESSAGE»;M$              (entrez le message)
30 FOR I=1 TO LEN(M$)
40 LET J=1-INT(1/5)*5+1
50 REM ***DÉCALAGE SELON LA CLEF
60 LET M=ASC(MID$(M$,I,1))-VAL(MID$(K$,J,1))
70 IF M<65 THEN LET M=M+26
80 PRINT CHR$(M);
90 NEXT I
```

Pour le Spectrum, la ligne 60 doit être remplacée par :

```
60 LET M=CODE(M$(I)) - VAL(K$(J))
```

Avec cette clef, le début de la phrase donne ceci une fois codé : ALQMM... Dans cet exemple, l'analyse de fréquence des lettres codées ne pourra donner aucun résultat puisque la valeur de décalage est différente pour chaque lettre. Une lettre correspondra à différentes lettres originelles selon sa place. Un autre chiffrage selon une autre méthode également simple donnera :

RECOS M'DAFINROTIQUE QUETAPRI UT AMIAFILLE

En y regardant de près, on s'aperçoit que cette chaîne de caractères codés est un anagramme de COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE, avec les espaces entre les mots. Nous pourrions déterminer un algorithme de codage en examinant le texte en clair et le texte codé. Cette méthode est très commune; si le texte chiffré doit être compréhensible par la destinataire, le mélange des lettres doit être cohérent. La méthode exposée, appelée *Bar Fence,* exige de la part du destinataire qu'il connaisse la clef. La clef est ici 3. Prenons le premier caractère et faisons-le suivre de trois espaces, prenons le suivant, jusqu'à la fin du message (l'apostrophe n'étant pas prise en compte).

C--S *---F---A---U-*--A---U-*--*F---L---E

Le reste s'obtient en remplissant les espaces sur plusieurs lignes au-dessous de la première ligne, tout en gardant un espace entre chaque lettre pour la ligne suivante, chaque ligne complétant la ligne précédente.

C S N M Q P I * F L E
O R * I F R A I U * R T Q E E * A I I L
U D O T E A U T M A

La lecture se fait de haut en bas, de biais, et ensuite de bas en haut, en prenant à chaque fois une seule lettre par ligne. Les étoiles représentent les blancs entre les mots.

Les méthodes exposées ici sont des exemples de chiffrement. Le chiffrement utilise la transformation ou la substitution des lettres selon une clef. Les codes sont différents dans la mesure où ils remplacent des blocs entiers de caractères par d'autres, généralement moins longs, ce qui rend également possible la compression des données. Leur inconvénient est d'exiger de l'émetteur et du destinataire qu'ils possèdent un livre donnant les codes. Un exemple de cette technique utilise un texte où les positions des mots du message sont codées. Prenons une phrase : « Au cours de la soirée, Julien voulait en finir avec sa leçon d'informatique. "Pas pratique de se rendre chez Charles en métro, je prends l'auto familiale", se dit-il. »

La clef est ici 2,11,2--. Cherchez le message!

Chaque chiffre indique la position du mot à retenir pour le message, à partir du premier mot du texte fictif, puis à partir du premier mot retenu pour le texte reconstitué.

Un ordinateur, de quelque type qu'il soit, apporte une aide prodigieuse pour le chiffrement ou le déchiffrement de messages. Pour le « chiffre de César », par exemple, il faut que l'ordinateur parcoure une chaîne alphanumérique, en ajoutant à chaque lettre une constante à la valeur ASCII correspondante; constante qui doit pouvoir changer lors de l'exécution et doit aussi boucler l'alphabet (le A est après le Z).

Cryptographie
Une des premières utilisations des ordinateurs fut de décrypter les codes de substitution à plusieurs clefs utilisés pendant la dernière guerre mondiale. Les Allemands avaient créé une machine qui générait son propre chiffre, Enigma. Les cryptogrammes qui en résultèrent furent extrêmement complexes et obligèrent les Alliés à consacrer une grande partie de leurs activités à la recherche de leur interprétation. Ils y parvinrent en partie grâce à Alan Turing qui, à Bletchley Park, travailla au sein d'un groupe dénommé Colossus. (Cl. Science Museum.)

Une question de style

Maintenant que nous avons vu les règles fondamentales du basic, nous pouvons examiner les aspects importants du style et de nouvelles commandes afin de perfectionner notre technique.

Le carnet d'adresses informatisé que nous avons développé dans les numéros précédents met en œuvre les traits les plus importants du langage BASIC, mais pas tous. A la fin de ce cours, nous verrons si vous voulez devenir un programmeur averti. Malheureusement, il est impossible d'être exhaustif, et le lecteur aura intérêt à se référer au manuel d'utilisation ou à l'un des nombreux ouvrages qui ont été publiés sur les ordinateurs domestiques, pour une analyse plus approfondie de la version de BASIC adaptée à sa machine.

Programmes en langage machine

La plupart des versions de BASIC comportent des routines écrites en langage machine. En gros, il y a deux manières de les inclure dans le programme. La plus simple consiste à utiliser PEEK et POKE. PEEK est une instruction pour examiner des adresses de mémoire spécifiques. Par exemple, LET X=PEEK(1000) prendra la valeur stockée à l'adresse 1000 pour l'attribuer à la variable X. En exécutant PRINT X, on imprimera la valeur qui était (et est toujours) localisée en 1000. Voici un petit programme pour jeter un coup d'œil (« peek ») sur les contenus de 16 adresses mémoire et les afficher à l'écran :

```
10 INPUT «ENTREZ L'ADRESSE DU PREMIER "PEEK" »; S
20 PRINT
30 FOR L=1 TO 16
40 LET A=PEEK(S)
50 PRINT «L'ADRESSE »;S;« CONTIENT: »;A
60 LET S=S+1
70 NEXT L
80 PRINT «APPUYEZ BARRE D'ESPACEMENT POUR
   EXAMINER LES 16 ADRESSES SUIVANTES»
90 PRINT «OU SUR "RETURN" POUR ARRÊTER»
100 FOR I=1 TO 1
110 LET C$=INKEY$
120 IF C$<>CHR$(13) AND C$<>«» THEN I=0
130 NEXT I
140 IF C$=CHR$(13) THEN GOTO 160
150 GOTO 30
160 END
```

La boucle aux lignes 100 à 130 vérifie la touche tapée sur le clavier et va à la fin du programme, si on a tapé RETURN (13 en ASCII), ou revient au commencement, en sautant l'instruction INPUT.

Si on le désire, le caractère ASCII en mémoire peut aussi être affiché en utilisant l'instruction PRINT CHR$(A). Mais attention! Les valeurs ASCII inférieures à 32 (en décimal) ne sont pas uniformément définies. Les valeurs ASCII de 0 à 31 représentent des caractères non affichables ou des fonctions spéciales, telles que les contrôles de curseur. 32 est le code pour « espace ». Le seul accord entre les différents constructeurs concerne le 13, qui est en général le code de retour de chariot, et le 7, qui commande le haut-parleur interne ou produit un « bip ».

POKE est l'inverse de PEEK. Il permet d'écrire n'importe quelle valeur, de 0 à 255, dans n'importe quelle adresse de RAM. Cette possibilité doit être utilisée avec précaution, car si l'on écrit dans une partie de mémoire déjà utilisée par le programme, on peut aboutir à des résultats inattendus, voire catastrophiques. Des routines écrites en langage machine peuvent être entrées au moyen de POKE à l'adresse appropriée et appelées au cours du programme par l'instruction CALL. Cela nous mènerait trop loin d'expliquer comment écrire des programmes en langage machine. Il suffit de dire que le langage machine est bien plus rapide que le meilleur des dialectes BASIC.

Le déplacement du curseur

Beaucoup d'ordinateurs domestiques permettent d'accéder directement à des positions de l'écran. Si ce n'est pas le cas de votre machine, il est néanmoins possible de déplacer le curseur vers la gauche, la droite, le haut et le bas de l'écran. Il faut d'abord connaître les codes ASCII représentant les touches de contrôle du curseur. Le petit programme suivant vous demande de taper une touche et vous donne ensuite la valeur ASCII qui lui correspond :

```
1 REM TROUVER LES CODES ASCII POUR LES TOUCHES
  DE CURSEUR
10 PRINT «TAPER UNE TOUCHE»;
20 FOR I=1 TO 1
30 LET K$=INKEY$
40 IF K$=«» THEN I=0
50 NEXT I
60 PRINT ASC(K$)
70 GOTO 10
80 END
```

Cette routine vous permettra aussi de trouver le code pour la touche RETURN (en général 13), ESC (caractère d'échappement — en général 27) et espace (en général 32), en plus des codes pour les touches de contrôle du curseur. L'ordinateur Sord M23, sur lequel ont été développés les programmes de ce cours, utilise les valeurs 8 pour le

curseur à gauche, 28 pour le curseur à droite, 29 en haut et 30 en bas. Votre ordinateur utilisera probablement d'autres valeurs. En substituant les valeurs que vous avez trouvées pour les codes de contrôle de curseur de votre ordinateur dans le programme ci-dessus, essayez :

```
10 PRINT CHR$(12): REM EFFACE L'ÉCRAN OU UTILISER LE
   CODE APPROPRIÉ
20 FOR L=1 TO 39
30 PRINT «*»;
40 NEXT L
50 FOR L=1 TO 22
60 PRINT CHR$(8);: REM CODE "CURSEUR A GAUCHE"
70 NEXT L
80 FOR L=1 TO 4
90 PRINT «@ »;
100 NEXT L
110 END
```

Voici ce qu'on devrait voir affiché :

```
**********@@@@**********
```

Les lignes 20 à 40 affichent simplement une ligne de 39 astérisques. Mais les lignes 50 à 70 « affichent » le « caractère curseur à gauche » 22 fois, de sorte que le curseur recule de 22 positions. Les lignes 80 à 100 affichent @ quatre fois et le programme se termine. De telles techniques de programmation permettent au programmeur de déplacer le curseur sur tout l'écran afin d'afficher de nouveaux caractères dans de nouvelles positions qui peuvent être inconnues jusqu'à ce que les valeurs soient calculées dans le programme. Cette technique a l'avantage de permettre d'utiliser des caractères d'écran ordinaires pour tracer des graphiques simples.

Pour voir comment ce type de contrôle de curseur peut servir à faire des graphes, essayez le petit programme suivant :

```
10 PRINT «CE PROGRAMME AFFICHE UN GRAPHIQUE
   A BARRES DE 3 VARIABLES»
20 INPUT «ENTREZ LES TROIS VALEURS »; X,Y,Z
30 PRINT
40 FOR L=1 TO 2
50 FOR A=1 TO X
60 PRINT «*»;
70 NEXT A
80 PRINT CHR$(13)
90 NEXT L
100 FOR L=1 TO 2
110 FOR A=1 TO Y
120 PRINT «*»;
130 NEXT A
140 PRINT CHR$(13)
150 NEXT L
160 FOR L=1 TO 2
170 FOR A=1 TO Z
180 PRINT «#»;
190 NEXT A
200 PRINT CHR$(13)
210 NEXT L
220 PRINT
230 END
```

Ce programme affiche un graphe à barres des trois variables. Les barres sont horizontales, partant de la gauche et suivant le mouvement « naturel » du curseur. Remarquez que l'on utilise PRINT CHR$(13) aux lignes 80, 140 et 200 car les points-virgules à la fin des instructions PRINT suppriment le retour de chariot (13 est le code ASCII pour RETURN).

Compléments sur les variables

Jusqu'ici, nous avons considéré les variables numériques et de chaîne. En fait, il y a plusieurs types de variables numériques reconnues par le BASIC, et un programmeur doit toujours spécifier le bon type pour économiser de la mémoire et assurer l'exactitude.

Lorsqu'une variable est déclarée dans un langage de programmation, un certain espace en mémoire sera aussitôt alloué à cette variable. Si le programme sait que la variable sera toujours un entier (par exemple LET SCORE=TOTAL+BONUS–PÉNALITÉ), il faudra réserver moins de mémoire. Si nous avons une variable qui peut prendre une infinité de valeurs différentes (par exemple LET SURFACE = PI*RAYON*RAYON), il faudra lui allouer plus d'espace.

En développant notre carnet d'adresses informatisé, nous nous sommes familiarisés avec la convention qui spécifie les variables de chaîne en faisant suivre le nom de variable par le signe $ (par exemple LET CLERECH$=MODCHP$(TAILLE)). On a supposé que les variables sans le signe « dollar » étaient des variables numériques ordinaires. Toutefois des conventions analogues peuvent être utilisées après les noms de variables pour spécifier le type de variable numérique.

Un nom de variable sans spécificateur est considéré comme une variable numérique réelle de simple précision. La plupart des BASIC reconnaissent les autres signes suivants : % pour spécifier une variable entière, ! pour une variable en simple précision et # en double précision (c'est-à-dire une variable qui peut stocker deux fois plus de chiffres significatifs). Voici un extrait d'un programme hypothétique qui utilise ces signes :

```
70 LET JOUEUR$=«JEAN»: REM VARIABLE DE CHAINE
80 LET SCORE%=0: REM VARIABLE ENTIÈRE
90 LET PI!=3.1416: REM VARIABLE EN SIMPLE PRÉCISION
100 LET SURFACE =PI*R*R: REM VARIABLE EN DOUBLE
    PRÉCISION
110 LET ESSAI=6: REM VARIABLE SUPPOSÉE RÉELLE EN
    SIMPLE PRÉCISION.
```

Tous les BASIC n'admettent pas tous ces types de variables. Le Spectrum, par exemple, n'a pas de variables entières. Les entiers sont simplement stockés comme des nombres réels en simple précision. Il n'admet pas non plus les nombres en double précision. Cependant les nombres en simple précision en BASIC Spectrum sont calculés avec neuf chiffres significatifs, contre sept seulement en BASIC Microsoft. Le BBC Micro admet les entiers et les réels en simple précision avec neuf chiffres significatifs. Le BASIC Microsoft comporte des variables en double précision avec 16 chiffres significatifs.

Les ordinateurs qui acceptent les variables entières attribuent généralement 2 octets au stockage du nombre, qui peut être compris entre —32,768 à 32,767. Cet intervalle est la plupart du temps tout à fait adapté à des variables telles que des scores, des nombres d'employés, des comptes de boucles FOR... NEXT et tous autres nombres susceptibles de ne prendre que des valeurs entières. Comme il ne faut que 2 octets pour stocker le nombre, en utilisant des variables entières s'il y a lieu, on économisera de la mémoire.

La dernière partie du cours de programmation BASIC traitera notamment des nombreux avantages et des quelques désavantages du BASIC comme langage.

Variantes de basic

INKEY$

Sur le Lynx, dans le premier programme, remplacez les lignes 110, 120 et 140 par :

```
110 C=KEYN
120 IF(C<> 13) AND (C<> 32) THEN I=0
140 IF C=13 THEN GOTO 160
```

CURSOR MOVEMENT

Le troisième programme tournera, mais ne produira pas le résultat escompté sur le Dragon, le BBC et le Lynx.

PEEK

Sur le BBC Micro, remplacez PEEK(S) par ?S.

Carnet d'adresses de Spectrum

Voici la version complète sur Spectrum du programme de carnet d'adresses informatisé. Les variantes de BASIC pour le Lynx, le Dragon 32, le BBC Micro, le Commodore 64 et le VIC-20 seront publiées dans le prochain numéro et se rapporteront à ce listage.

```
1 REM *CRÉATION D'UN FICHIER DE DONNÉES*
2 DIM N$(50,30)
3 LET N$(1) = «@PREMIER»
4 INPUT «INSÉREZ LA BANDE DE DONNÉES, APPUYEZ SUR LA
  TOUCHE 'PLAY' ET TAPEZ 'ENTER'»;A$
5 SAVE «NCHP» DATA N$()
6 INPUT «REMBOBINEZ LA BANDE, APPUYEZ SUR LA TOUCHE
  'PLAY', TAPEZ 'ENTER'»;A$
7 VERIFY «NCHP» DATA N$()
8 STOP
```

C'est le programme d'initialisation qui crée le tableau sur la bande pour la première fois. Quand vous aurez fait tourner ce programme, rembobinez la bande Data, chargez le programme principal (listage ci-dessous) et faites RUN. Vous n'aurez pas besoin de réinitialiser le programme, à moins que vous vouliez créer un nouveau fichier d'adresses.

```
10 REM «PROGRAMME PRINCIPAL»
20 REM *INITIL*
30 GOSUB 1000
40 REM *ACCUEIL*
50 GOSUB 3000
60 FOR M = 1 TO 1
70 LET M = 0
80 REM*CHOIX*
90 GOSUB 3500
100 REM *EXECUT*
110 GOSUB 4000
120 IF CHOI = 9 THEN LET M = 1
130 NEXT M
140 STOP

1000 REM S—P *INITIL*
1010 GOSUB 1100
1020 GOSUB 1400
1030 GOSUB 1600
1090 RETURN

1100 REM S—P *CRETAB*
1110 DIM N$(50,30)
1120 DIM M$(50,30)
1130 DIM S$(50,30)
1140 DIM T$(50,15)
1150 DIM C$(50,15)
1160 DIM R$(50,15)
1170 DIM X$(50,30)
1180 DIM B$(30):DIM Z$(30)
1190 DIM U$(30):DIM W$(15)
1210 LET TAILLE = 0
1220 LET RMOD = 0
1230 LET TRIE = 1
1240 LET ACT = 0
1250 LET Z$ = «@PREMIER»
1260 LET Q$ = B$
1300 RETURN

1400 REM S—P *LECFCR*
1405 INPUT «INSÉREZ LA BANDE DE DONNÉES, APPUYEZ SUR
  LA TOUCHE 'PLAY' ET TAPEZ 'ENTER'»;A$
1410 LOAD «NCHP» DATA N$()
1420 IF N$(1) = Z$ THEN LET Q$ = Z$: RETURN
1430 LOAD «MCHP» DATA M$()
1440 LOAD «SCHP» DATA S$()
1450 LOAD «TCHP» DATA T$()
1460 LOAD «CCHP» DATA C$()
1470 LOAD «TELCHP» DATA R$()
1480 LOAD «NDXCHP» DATA X$()
1485 INPUT «ARRETEZ LA BANDE ET TAPEZ 'ENTER'»;A$
1490 REM 'TAILLEFCR'
1500 LET TAILLE = 51
1510 FOR  L = 1 TO 50
1520 IF N$(L) = B$ THEN LET TAILLE = L: LET L = 50
1530 NEXT L
1540 RETURN

1600 REM S—P *DEFDRA*
1640 IF Q$ = Z$ THEN LET TAILLE = 1
1690 RETURN

3000 REM S—R *ACCUEIL*
3010 CLS
3020 PRINT:PRINT:PRINT:PRINT
3060 PRINT TAB(12);«VOICI LE PROGRAMME»
3070 PRINT TAB(4);«DE CARNET D'ADRESSES INFORMATISÉ»
3080 PRINT TAB (10);«DE ABC INFORMATIQUE»
3090 PRINT
3100 PRINT TAB (1);«(APPUYEZ SUR LA BARRE D'ESPACEMENT
  POUR CONTINUER)»
3110 FOR L = 1 TO 1
3120 IF INKEY$<>« » THEN LET L = 0
3130 NEXT L
3140 CLS
3150 RETURN

3500 S—R *CHOIX*
3520 IF Q$ = Z$ THEN GOSUB 3860: RETURN
3540 REM 'CHMENU'
3550 CLS
3560 PRINT «CHOISISSEZ UNE DES OPTIONS SUIVANTES»
3570 PRINT:PRINT:PRINT
3600 PRINT «1. TROUVER UN ENREGISTREMENT (A PARTIR
  D'UN NOM)»
3610 PRINT «2. TROUVER DES NOMS (A PARTIR D'UN NOM
  INCOMPLET)»
3620 PRINT «3. TROUVER DES ENREGISTREMENTS (A PARTIR
  D'UNE VILLE)»
3630 PRINT «4. TROUVER DES ENREGISTREMENTS (A PARTIR
  D'INITIALES)»
3640 PRINT «5. LISTER TOUS LES ENREGISTREMENTS»
3650 PRINT «6. AJOUTER UN NOUVEL ENREGISTREMENT»
3660 PRINT «7. MODIFIER UN ENREGISTREMENT»
3670 PRINT «8. EFFACER UN ENREGISTREMENT»
3680 PRINT «9. QUITTER ET SAUVEGARDER»
3690 PRINT:PRINT
3710 REM 'ENTCHOI'
3750 PRINT «ENTRER UNE OPTION (1 A 9)»
3760 FOR L = 1 TO 1
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = «» THEN I = 0
3800 NEXT I
3810 LET CHOI = CODE A$—48
3820 IF (CHOI<1) OR (CHOI >9) THEN LET L = 0
3840 NEXT L
3850 RETURN

3860 REM S—P *PREM*
3870 LET CHOI = 6
3880 CLS
3890 PRINT
3900 PRINT TAB(5);«IL N'Y A PAS D'ENREGISTREMENTS»
3910 PRINT TAB(6);«DANS LE FICHIER. VOUS DEVREZ»
3920 PRINT TAB(4);«D'ABORD AJOUTER UN ENREGISTREMENT»
3930 PRINT
3940 REM *CONTINUER*
3950 GOSUB 3100
3990 RETURN

4000 REM S—P *EXECUT*
4040 IF CHOI = 1 THEN GOSUB 5700
4050 REM 2 EST *TRNOMS*
4060 REM 3 EST *TRVILLE*
4070 REM 4 EST *TRINIT*
4080 REM 5 EST *LSTENR*
4090 IF CHOI = 6 THEN GOSUB 4200
4100 IF CHOI = 7 THEN GOSUB 6600
4110 IF CHOI = 8 THEN GOSUB 7500
4120 IF CHOI = 9 THEN GOSUB 5000
4140 RETURN

4200 REM S—P *AJOUTENR*
4210 CLS
4220 INPUT «ENTREZ LE NOM»;N$(TAILLE)
4230 INPUT «ENTREZ LA RUE»;S$(TAILLE)
4240 INPUT «ENTREZ LE CODE POSTAL»;C$(TAILLE)
4250 INPUT «ENTREZ LA VILLE»;T$(TAILLE)
4260 INPUT «ENTREZ LE NUMÉRO DE TÉLÉPHONE»;R$(TAILLE)
4270 LET RMOD = 1:LET TRIE = 0
4280 LET X$(TAILLE) = STR$(TAILLE)
4290 LET Q$ = «»
4300 GOSUB 4500
4310 LET CHOI = 0
4320 LET TAILLE = TAILLE + 1
4350 RETURN

4500 REM S—P *MODNOM*
4510 REM DÉPLACE A LA CASE SUPÉRIEURE
4520 LET D$ = N$(TAILLE):LET P$ = «»
4530 FOR L = 1 TO LEN(D$)
```

```
4540 LET A$ = D$(L)
4550 LET T =  CODE A$
4560 IF T >= 97 THEN LET T = T—32
4570 LET A$ = CHR$ T
4580 LET P$ = P$ + A$
4590 NEXT L
4600 LET D$ = P$: LET P$ = «»: LET A$ = «»: LET T = LEN(D$):
     LET S = 0
4610 REM LOCALISE LE PREMIER ESPACE
4620 FOR L = 1 TO T
4630 IF D$(L) = « » THEN LET S = L: LET L = T
4640 NEXT L
4650 REM ENLÈVE LE RESTE, MET LE PRÉNOM DANS P$
4660 FOR L = 1 TO S—1
4670 IF CODE(D$(L))>64 THEN LET P$ = P$ + D$(L)
4680 NEXT L
4690 REM ENLÈVE LE RESTE, MET LE NOM DANS A$
4700 FOR L = S + 1 TO LEN(D$)
4710 IF CODE(D$(L))>64 THEN LET A$ = A$ + D$(L)
4720 NEXT L
4730 LET M$(TAILLE) = A$ + « » + P$
4740 LET P$ = «»: LET A$ = «»: LET S = 0
4750 RETURN

5000 REM S—P *QUITTE*
5010 IF (RMOD = 0) AND (TRIE = 1) THEN RETURN
5020 IF (RMOD = 1) AND (TRIE = 0) THEN GOSUB 5200
5030 GOSUB 5600
5040 RETURN

5200 REM S—P *TRIENR*
5210 FOR K = 1 TO 1
5220 LET S = 0
5230 FOR L = 1 TO TAILLE—2
5240 LET T = L + 1
5250 IF M$(L)>M$(T) THEN GOSUB 5400
5260 NEXT L
5270 IF S = 1 THEN LET K = 0
5280 NEXT K
5290 LET TRIE = 1
5300 RETURN

5400 REM S—P *ECHENR*
5410 LET U$ = N$(L): LET N$(L) = N$(T): LET N$(T) = U$
5420 LET U$ = M$(L): LET M$(L) = M$(T): LET M$(T) = U$
5430 LET U$ = S$(L): LET S$(L) = S$(T): LET S$(T) = U$
5440 LET W$ = T$(L): LET T$(L) = T$(T): LET T$(T) = W$
5450 LET W$ = C$(L): LET C$(L) = C$(T): LET C$(T) = W$
5460 LET W$ = R$(L): LET C$(L) = C$(T): LET C$(T) = W$
5470 LET X$(L) = STR$(L)
5480 LET X$(T) = STR$(T)
5490 LET S = 1
5500 RETURN

5600 REM S—P *SAUVENR*
5605 INPUT «INSÉREZ LA BANDE ET TAPEZ 'ENTER'»;A$
5610 SAVE «NCHP» DATA N$()
5620 SAVE «MCHP» DATA M$()
5630 SAVE «SCHP» DATA S$()
5640 SAVE «TCHP» DATA T$()
5650 SAVE «CCHP» DATA C$()
5660 SAVE «TELCHP» DATA R$()
5670 SAVE «NDXCHP» DATA X$()
5680 INPUT «ARRETEZ LA BANDE ET TAPEZ 'ENTER'»;A$
5690 RETURN

5700 REM S—P *TROUVENR*
5710 CLS
5720 IF TRIE = 0 THEN GOSUB 5200
5730 PRINT:PRINT
5740 PRINT TAB(4);«TROUVER UN ENREGISTREMENT»
5750 PRINT TAB(8);«A PARTIR DU NOM»
5760 PRINT
5770 PRINT TAB(4);«TAPER LE NOM COMPLET»
5780 PRINT TAB(3);«EN COMMENCANT PAR LE PRÉNOM»
5790 PRINT:PRINT
5800 INPUT «LE NOM EST »;N$(TAILLE)
5810 GOSUB 4500
5820 LET U$ = M$(TAILLE)
5830 LET INF = 1
5840 LET SUP = TAILLE—1
5850 FOR X = 1 TO 1
5860 LET MD = INT((INF + SUP)/2)
5870 IF M$(MD)<>U$ THEN LET X = 0
5880 IF M$(MD)<U$ THEN LET INF = MD + 1
5890 IF M$(MD)>U$ THEN LET SUP = MD—1
5900 IF INF>SUP THEN LET X = 1
5910 NEXT X
5920 IF INF>SUP THEN LET ACT = 0
5930 IF INF<= SUP THEN LET ACT = MD
5940 IF ACT = 0 THEN GOSUB 6400:RETURN
5950 CLS
5960 PRINT
5970 PRINT TAB(5);«ENREGISTREMENT TROUVÉ»
5980 PRINT
5990 PRINT «NOM :»,N$(ACT)
6000 PRINT «RUE :»,S$(ACT)
6010 PRINT «CODE POSTAL :»,C$(ACT)
6020 PRINT «VILLE :»,T$(ACT)
6030 PRINT «NO DE TÉL.:»,R$(ACT)
6040 PRINT
6050 PRINT «TAPER UNE LETTRE POUR IMPRIMER»
6060 PRINT «OU LA BARRE D'ESPACEMENT POUR CONTINUER»
6070 FOR I = 1 TO 1
6080 LET A$ = INKEY$
6090 IF A$ = « » THEN LET I = 0
6100 NEXT I
6110 IF A$<>« » THEN GOSUB 6200
6120 RETURN

6200 REM S—R *LISTACT*
6210 LPRINT
6220 LPRINT «NOM»,N$(ACT)
6230 LPRINT «RUE»,S$(ACT)
6240 LPRINT «CODE POSTAL»,C$(ACT)
6250 LPRINT «VILLE»,T$(ACT)
6260 LPRINT «NO DE TÉL.»,R$(ACT)
6270 LPRINT:LPRINT
6280 RETURN

6400 REM S—P *PASENR*
6410 CLS
6420 PRINT TAB(4);«*ENREGISTREMENT NON TROUVÉ*»
6430 PRINT TAB(2);«*SOUS LA FORME»;N$(TAILLE);«*»
6440 PRINT
6450 REM 'CONTINUER'
6460 GOSUB 3100
6470 RETURN

6600 REM S—P *MODENR*
6610 CLS
6620 PRINT:PRINT:PRINT
6630 LET E$ = CHR$ 13
6640 PRINT TAB(3);«*POUR MODIFIER UN ENREGISTREMENT*»
6650 PRINT TAB(2);«*LOCALISEZ D'ABORD CET
     ENREGISTREMENT*»
6660 GOSUB 5720
6670 IF ACT = 0 THEN RETURN
6680 PRINT
6690 PRINT TAB(2);«VOULEZ-VOUS MODIFIER LE NOM ?»
6700 PRINT
6710 PRINT «TAPER 'ENTER' POUR ENTRER UN NOUVEAU NOM»
6720 PRINT «OU LA BARRE D'ESPACEMENT POUR LA ZONE
     SUIVANTE»
6730 FOR I = 1 TO 1
6740 LET A$ = INKEY$
6750 IKF (A$<>E$) AND (A$<>« ») THEN LET I = 0
6760 NEXT I
6770 IF A$ = E$ THEN INPUT «NOUVEAU NOM»;N$(ACT)
6780 IF A$ = E$ THEN LET RMOD = 1
6790 IF A$ = E$ THEN TRIE = 0
6800 IF A$ = E$ THEN LET N$(TAILLE) = N$(ACT)
6810 IF A$ = E$ THEN GOSUB 4500
6820 IF A$ = E$ THEN LET M$(ACT) = M$(TAILLE)
6830 PRINT
6840 PRINT «VOULEZ-VOUS MODIFIER LA RUE ?»
6850 PRINT
6860 PRINT «TAPEZ 'ENTER' POUR ENTRER LA NOUVELLE RUE»
6870 PRINT «OU LA BARRE D'ESPACEMENT POUR LA ZONE
     SUIVANTE»
6880 FOR I = 1 TO 1
6890 LET A$ = INKEY$
6900 IF(A$<>E$) AND (A$<>« ») THEN LET I = 0
6910 NEXT I
6920 IF A$ = E$ THEN LET RMOD = 1
6930 IF A$ = E$ THEN INPUT «NOUVELLE RUE:»;S$(ACT)
6940 PRINT
6950 PRINT «VOULEZ-VOUS MODIFIER LA VILLE ?»
6960 PRINT
6970 PRINT «TAPEZ 'ENTER' POUR ENTRER LA NOUVELLE
     VILLE»
6980 PRINT«OU LA BARRE D'ESPACEMENT POUR LA ZONE
     SUIVANTE»
6990 FOR I = 1 TO 1
7010 LET A$ = INKEY$
7020 IF(A$<>E$) AND (A$<>« ») THEN LET I = 0
7030 NEXT I
7040 IF A$ = E$ THEN LET RMOD = 1
7050 IF A$ = E$ THEN INPUT «NOUVELLE VILLE»;T$(ACT)
7060 PRINT
7070 PRINT«VOULEZ-VOUS MODIFIER LE CODE POSTAL ?»
7080 PRINT
7090 PRINT «TAPEZ 'ENTER' POUR ENTRER LE NOUVEAU
     CODE POSTAL»
7100 PRINT «OU LA BARRE D'ESPACEMENT POUR LA ZONE
     SUIVANTE»
7110 FOR I = 1 TO 1
7120 LET A$ = INKEY$
7130 IF (A$<>E$) AND(A$<>« ») THEN LET I = 0
7140 NEXT I
7150 IF A$ = E$ THEN LET RMOD = 1
7160 IF A$ = E$ THEN INPUT «NOUVEAU CODE POSTAL»;C$(ACT)
7170 PRINT
7180 PRINT «VOULEZ-VOUS MODIFIER LE NO DE TÉLÉPHONE ?»
7190 PRINT
7200 PRINT «TAPEZ 'ENTER' POUR ENTRER LE NOUVEAU NO
     DE TÉL.»
7210 PRINT «OU LA BARRE D'ESPACEMENT POUR CONTINUER»
7220 FOR I = 1 TO 1
7230 LET A$ = INKEY$
7240 IF (A$<>E$) AND (A$<>« ») THEN LET I = 0
7250 NEXT I
7260 IF A$ = E$ THEN LET RMOD = 1
7270 IF A$ = E$ THEN INPUT «NOUVEAU NUMÉRO »;R$(ACT)
7280 RETURN

7500 REM S—P *EFFENR*
7510 CLS
7520 PRINT:PRINT:PRINT:PRINT
7530 LET E$ = CHR$ 13
7540 PRINT TAB(3);«*POUR EFFACER UN ENREGISTREMENT*»
7550 PRINT TAB(2);«*LOCALISEZ D'ABORD CET
     ENREGISTREMENT*»
7560 GOSUB 5720
7570 IF ACT = 0 THEN RETURN
7580 PRINT
7590 PRINT «VOULEZ-VOUS EFFACER CET ENREGISTREMENT ?»
7600 PRINT «* ATTENTION *... C'EST VOTRE DERNIÈRE
     CHANCE !»
7610 PRINT
7620 PRINT «TAPER 'ENTER' POUR EFFACER»
7630 PRINT «OU LA BARRE D'ESPACEMENT POUR CONTINUER»
7640 FOR I = 1 TO 1
7650 LET A$ = INKEY$
7660 IF (A$<>E$) AND(A$<>« ») THEN LET I = 0
7670 NEXT I
7680 IF A$ = « » THEN RETURN
7690 FOR L = ACT TO TAILLE—2
7700 LET T = L + 1
7710 LET N$(L) = N$(T)
7720 LET M$(L) = M$(T)
7730 LET S$(L) = S$(T)
7740 LET T$(L) = T$(T)
7750 LET C$(L) = C$(T)
7760 LET R$(L) = R$(T)
7770 LET X$(L) = X$(T)
7780 NEXT L
7790 LET RMOD = 1
7800 LET TAILLE = TAILLE—1
7810 RETURN
```

Tony Lodge

Le défi universitaire

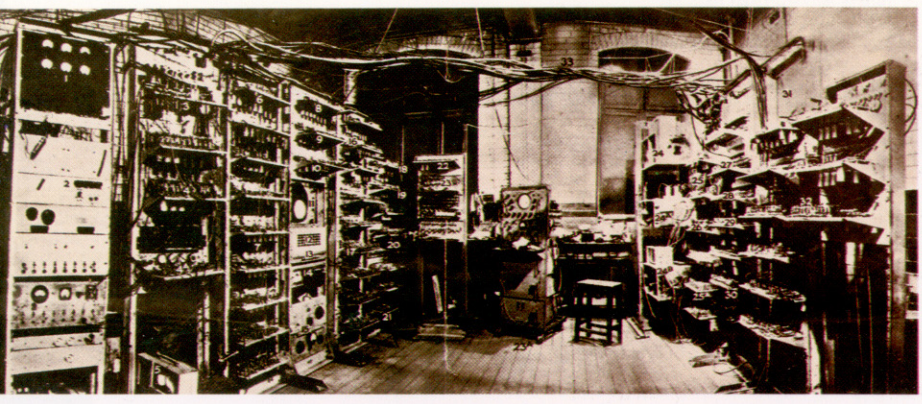
Le premier ordinateur programmable du monde fut mis au point à l'université de Manchester.

Après la fin de la Seconde Guerre mondiale, l'université de Manchester engagea deux nouveaux enseignants. Max Newman, qui avait cassé, à Bletchley Park, le code militaire allemand (avec Colossus, le premier ordinateur électromécanique) y devint professeur de mathématiques. La direction de la section du génie électrique revint à un ingénieur radar, F.C. Williams, qui amena avec lui un jeune assistant nommé Tom Kilburn. Celui-ci était familier des appareils à mémoire électronique à impulsions, car il les avait utilisés durant la guerre lors de ses travaux sur les radars.

En 1946, lors d'une tournée des bases radar américaines, on montra à Williams le prototype de l'ENIAC, l'ordinateur à tubes. Une fois rentré, il parvint à convaincre la Royal Society d'investir 35 000 livres dans la création à Manchester d'un laboratoire consacré à la mise au point d'une machine à calculer. L'université de Manchester n'était pas seule à vouloir réaliser un ordinateur programmable. Celle de Pennsylvanie construisait l'EDVAC, celle de Cambridge travaillait sur l'EDSAC, et le National Physical Laboratory poursuivait la réalisation de l'ACE. Tous ces projets utilisaient une mémoire à lignes à retard (tubes de mercure). A Manchester, on décida de construire un appareil dont la mémoire, due à Williams, faisait appel à un tube cathodique. Dès l'automne 1947, on était déjà parvenu à obtenir 2 048 bits, pendant plusieurs heures.

Grâce au « tube de Williams », le Mark I parvint à faire tourner un programme en juin 1948, et devint le premier ordinateur programmable du monde. Il pouvait exécuter une instruction en 1,2 milliseconde. Le tube cathodique permettait d'obtenir une mémoire à accès direct; le contenu de la mémoire principale de l'unité de contrôle pouvait être affiché sur écran.

Manchester Mark I
L'appareil put exécuter un programme dès juin 1948; on peut y voir le premier ordinateur numérique du monde. Ferranti, qui était alors une simple firme locale, fut chargé d'en assurer la commercialisation dès le début de 1951.
(Cl. université de Manchester.)



Une fois démontrées les possibilités du tube de Williams pour le stockage des informations, on construisit un modèle amélioré du Mark I qui pouvait travailler sur les nombres premiers, ou sur des problèmes d'optique. Le principal responsable scientifique du gouvernement fut si impressionné par les performances de l'appareil qu'il intervint pour qu'une firme de Manchester en assure la fabrication commerciale. Produit par Ferranti, le Mark I fut mis sur le marché en février 1951, battant l'UNIVAC de cinq mois et devenant le premier ordinateur du monde commercialement disponible.

Sur ce nouveau modèle, les instructions pouvaient être modifiées en cours d'exécution grâce à une mémoire supplémentaire appelée tube B. Au moment voulu, ce dispositif faisait passer dans le registre de contrôle le contenu de sa propre mémoire, ce qui avait pour résultat de modifier le programme original et d'en accélérer l'exécution. Certains brevets de l'université de Manchester furent rachetés par IBM lorsque cette société mit au point ses premiers ordinateurs. Comme il visitait un jour le quartier général de la firme, où l'on pouvait lire partout la célèbre devise « Pensez », Williams se vit demander comment les gens de Manchester avaient réussi à construire un tel appareil, alors que toutes les ressources d'IBM n'avaient pas suffi. « Nous n'avons pas cessé de *trop* y penser! » répondit-il aussitôt.

Alan Turing arriva à Manchester en 1948, ce qui eut un effet stimulant sur la recherche; deux ans plus tard il avait rédigé le premier manuel de programmation. En 1952, il fut décidé de construire une machine plus compacte et plus économique. L'invention du transistor accéléra les choses, et en novembre 1953 le premier ordinateur utilisant cette nouvelle technologie devint opérationnel.

A la fin des années cinquante, les Américains reprirent la tête dans le domaine de la recherche informatique. Le gouvernement britannique décida de financer un projet qui rendrait à la Grande-Bretagne sa suprématie. L'ordinateur Atlas, construit sous la direction de Tom Kilburn, fut officiellement commandé en décembre 1962. Il utilisait des mots de 48 bits à une adresse, disposait d'une mémoire principale de 16 K, et d'une mémoire morte à tambour de 8 K. Plusieurs années durant, il fut considéré comme le plus avancé de tous les ordinateurs existants. Presque au même moment, en France, d'autres pionniers faisaient parler d'eux avec les ordinateurs Bull.

PROGRAMME N° 11

DÉCHAINEZ-VOUS!

TRAITEMENT DES CHAINES

Jusqu'à présent, les seules variables utilisées furent des nombres ou des chiffres, mais la machine peut aussi manipuler des variables, lettres ou symboles. Elle peut traiter des caractères séparés ou des chaînes entières. Ces variables appelées « alphanumériques » comportant des chaînes de caractères ont un nom suivi du symbole $ (dollar).

Quelques exemples de variable alphanumérique :

A$ B$ NM$

La variable Z est différente de la variable Z$ et toutes deux peuvent figurer dans le même programme, exemple :

```
Z = 10
Z$ = « NAPOLÉON »
```

Les caractères intégrés dans une variable alphanumérique s'écrivent entre guillemets.

L'instruction : ? Z$ imprime le contenu de la variable Z$.

Ainsi, vous pouvez mémoriser dans une variable dont le nom est très court une chaîne de caractères que vous utilisez souvent :

INSTRUCTION « LEN »

Il existe d'autres instructions traitant des chaînes. Pour connaître la longueur d'une chaîne (le nombre de caractères qu'elle contient) vous pouvez taper :

```
? LEN (NAPOLÉON)
```

LEN représente LENGTH (« longueur »)

La machine affichera ici 8.

Remarque. Les espaces sont considérés comme des caractères. Le nombre de caractères d'une chaîne varie entre 0 et 225.

Pour ne faire imprimer qu'une partie d'une chaîne de caractères on peut utiliser 3 instructions :

LEFT$ RIGHT$ MID$

Supposons Z$ = « NAPOLÉON BONAPARTE »

Si vous voulez imprimer les 8 premières lettres de Z$, il vous suffit de taper :

```
? LEFT$ (Z$, 8)
```

et NAPOLÉON apparaît à l'écran.

Si vous tapez :

```
? RIGHT$ (Z$, 9)
```

Il apparaît BONAPARTE.

A chaque fois que vous écrivez un programme utilisant des variables alphanumériques, il vous faut déterminer la valeur de la variable.

A chaque exécution d'un programme, le système attribue la valeur 0 à toutes les variables numériques et la chaîne nulle aux variables alpha.

Voici un petit programme utilisant LEN et LEFT$

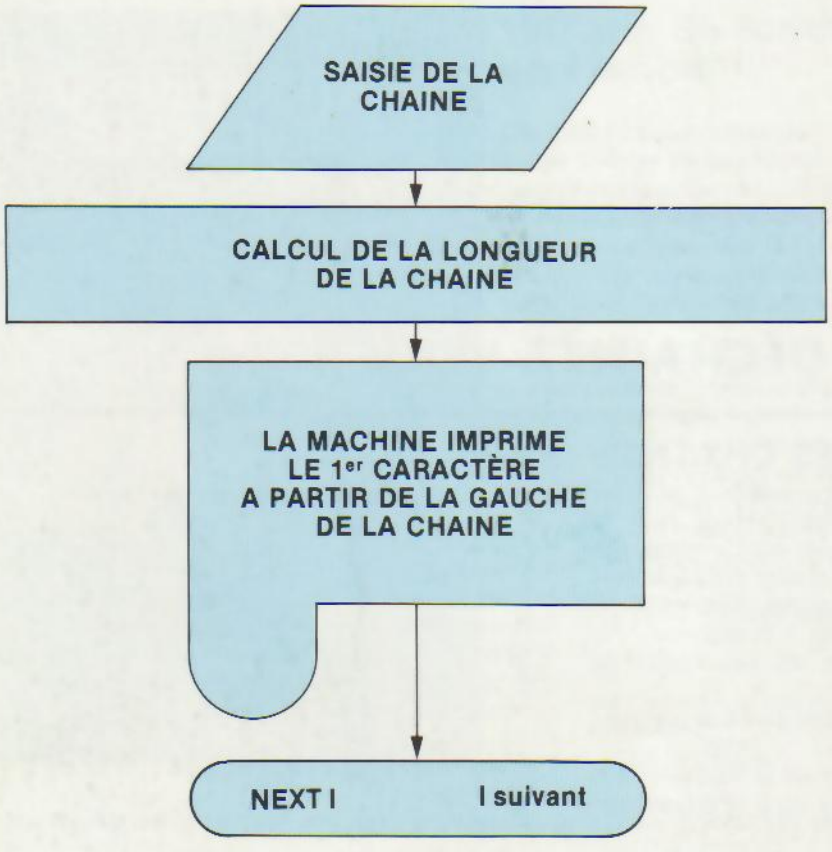
```
10 HOME
20 Z$ = "NAPOLEON BONAPARTE"
30 FOR K = 1 TO LEN (Z$)
40 PRINT LEFT$ (Z$,K)
50 NEXT K
§
§
§RUN
N
NA
NAP
NAPO
NAPOL
NAPOLE
NAPOLEO
NAPOLEON
NAPOLEON
NAPOLEON B
NAPOLEON BO
NAPOLEON BON
NAPOLEON BONA
NAPOLEON BONAP
NAPOLEON BONAPA
NAPOLEON BONAPAR
NAPOLEON BONAPART
NAPOLEON BONAPARTE
```

Vous pouvez remplacer la ligne 20 par

```
20 INPUT « Z$ = »; Z$
```

et ainsi saisir vous-même la chaîne de caractères désirée.

ORGANIGRAMME



Vous pouvez écrire un autre programme en remplaçant LEFT$ par RIGHT$

```
10 HOME
20 Z$ = "NAPOLEON BONAPARTE"
30 FOR K = 1 TO LEN (Z$)
40 PRINT RIGHT$ (Z$,K)
50 NEXT K
§
§
§RUN
E
ET
RTE
ARTE
PARTE
APARTE
NAPARTE
ONAPARTE
BONAPARTE
 BONAPARTE
N BONAPARTE
ON BONAPARTE
EON BONAPARTE
LEON BONAPARTE
OLEON BONAPARTE
POLEON BONAPARTE
APOLEON BONAPARTE
NAPOLEON BONAPARTE
```

Supposons maintenant que vous vouliez écrire les caractères « LÉON » de la chaîne Z$.
Il faut pour cela utiliser l'instruction MID$ en écrivant

PRINT MID$ (Z$,5,4)

Le premier nombre (5) indique le caractère de la chaîne où la machine commencera l'impression, le second, ici (4) déterminera le nombre de caractères à imprimer. Le système interprète la ligne d'instruction de la manière suivante :
1) localiser le 5e caractère de Z$ et imprimer 4 caractères en commençant par le 5e et en se déplaçant vers la droite.

Voici un petit programme pour utiliser à nouveau l'instruction MID$.

```
10 HOME
20 Z$ - "NAPOLEON"
30 PRINT : PRINT "TAPER UN NUMERO
           DE 1 A";LEN(Z$)
40 PRINT : PRINT "ET JE VOUS DONNERAI
           LA LETTRE DE"
50 PRINT : "NAPOLEON QUI Y CORRESPOND"
60 PRINT : INPUT "NUMERO:";N
70 IF N    LEN (Z$) OR N  1 GOTO 60
80 PRINT : PRINT MID$ (Z$,N,1);
           "EST LA LETTRE NUMERO",N
90 END
§
§
§RUN
TAPER UN NUMERO DE 1 A 8
ET JE VOUS DONNERAI LA LETTRE DE
NAPOLEON QUI Y CORRESPOND
NUMERO :5
L EST LA LETTRE NUMERO 5
§
§
§RUN
TAPER UN NUMERO DE 1 A 8
ET JE VOUS DONNERAI LA LETTRE DE
NAPOLEON QUI Y CORRESPOND
NUMERO :8
N EST LA LETTRE NUMERO 8
§
§
§RUN
TAPER UN NUMERO DE 1 A 8
ET JE VOUS DONNERAI LA LETTRE DE
NAPOLEON QUI Y CORRESPOND
NUMERO :10
NUMERO :0
NUMERO :6
E EST LA LETTRE NUMERO 6
```